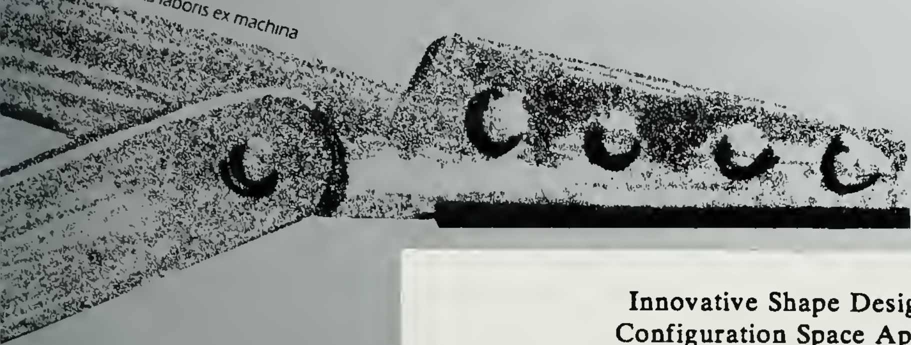
Robotics Research Technical Report

Generatorium omnis laboris ex machina



Innovative Shape Design: A Configuration Space Approach

by

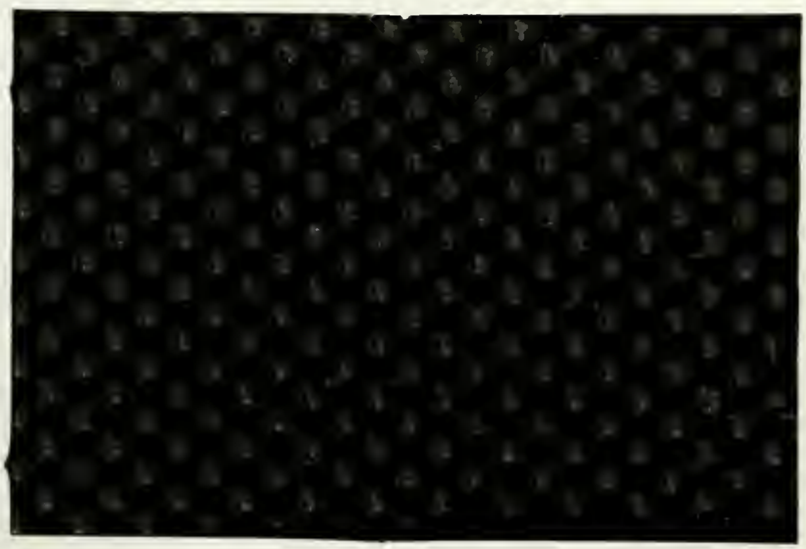
Leo Joskowicz
Sanjaya Addanki

Technical Report No. 356
Robotics Report No. 144
March, 1988

NYU COMPSCI TR-356 c.1
Joskowicz, Leo
Innovative shape design

New York University
Courant Institute of Mathematical Sciences

Computer Science Division
251 Mercer Street New York, N.Y. 10012



Innovative Shape Design: A Configuration Space Approach

by

Leo Joskowicz
Sanjaya Addanki

Technical Report No. 356
Robotics Report No. 144
March, 1988

New York University
Dept. of Computer Science
Courant Institute of Mathematical Sciences
251 Mercer Street
New York, New York 10012

Work on this paper has been supported by National Science Foundation Grant DCR-8603758.

Innovative Shape Design: A Configuration Space Approach

Leo Joskowicz
Department of Computer Science
Courant Institute of Mathematical Sciences
New York University
251 Mercer Street, New York, NY 10012

Sanjaya Addanki
IBM T.J. Watson Research Center
P.O. Box 704
Yorktown Heights, NY 10598

Abstract

In this paper, we address the problem of designing the shape of physical objects defined by a set of functional requirements. In particular, we show how to design elementary components of mechanical devices (kinematic pairs) from a description of their desired behavior. We begin by presenting two ways of describing kinematic behavior, a *possible motions* description and a *causal* description, and establish the correspondence between these descriptions and their equivalent two-dimensional configuration space representation. We then present a backtracking algorithm that modifies (or creates) object shapes by adding and deleting line segments and arcs to the objects' contours. The additions and deletions are guided by the configuration space description of the desired behavior. In the case of the design of two convex objects, we show that this algorithm has linear complexity. A number of efficient algorithms for the case of two translating objects are provided. In the last sections, we show how to deal with design problems in which the desired behavior is described in qualitative and causal terms. This work is based on the first principles theory of shape and kinematic function developed in a previous work [Joskowicz87c].

The first author was partially supported by a National Science Foundation grant under contract DCR-8603758 and by the Defense Advanced Research Projects Agency under contract N00014-85-K-0163 from the Office of Naval Research.

1 Introduction

Current theories of automated design identify two classes of design: routine design and innovative design ([Brown86], [Mittal86b], [Mitchell85]). In routine design, the design process starts from a generic specification of the desired device and progressively refines this specification until the detailed composition of the desired device is found. At each refinement step, the constraints generated by the new requirements guide the design process. The final step consists of choosing the appropriate elementary (or primitive) components that will form the device, together with the values of their parameters that satisfy the set requirements. Elementary components are chosen from a predefined and fixed library of components. Innovative design, on the other hand, does not presuppose an initial generic device specification nor a fixed library of components. It generally requires the ability to reason about the structure and underlying physics (first principles) of the domain of application, together with a set of design guidelines.

The scope of routine design depends directly on the modeling of the elementary components of the library. Elementary components are described functionally, i.e. their behavior is described by a set of parameters and relations that are considered appropriate for the domain of application. When the design specifications requires the consideration of an additional parameter, or the introduction of a new component, the design process will fail. Innovative design, on the other hand is only limited, in principle, by the adequacy of the domain model and the algorithms that manipulate it. Innovative design tends to be computationally more expensive than routine design, and is necessary only when explicit reasoning about the relation between structure and function is required. In certain domains, such as digital circuits, routine design is an appropriate design framework since most of the design is *structural design*. Elementary components (at the level of resistors, capacities, etc.) are generally simple and the modification or introduction of new basic elements is very seldomly considered. In other domains, such as mechanical devices or beam structure design, the modification and introduction of new basic components is often necessary to meet the desired requirements. In these domains, *component design* is pervasive, and thus requires an adequate first principles theory to create and modify basic components.

This paper addresses the problem of designing elementary components in the domain of mechanical devices. The automatic design of mechanical devices presents a number of interesting issues, not encountered in other domains (see [Dixon86]). The role of geometry, for example, is of primary importance in mechanisms. The motion of each object and the relations between object motions in the mechanism (the *kinematic behavior* of the mechanism) are directly determined by the shapes of the objects and the nature of the contacts between them. Elementary components are constituted by *pairs* of objects, rather than by individual objects ([Reuleux76]). Basic components are called *kinematic pairs*. Common examples of kinematic pairs are a screw and a bolt, a pair of meshed gears, prismatic joints, etc. Complex mechanisms are designed by assembling kinematic pairs that achieve the desired behavior.

It is a common observation that in order to comply with the given design requirements, new or modified shapes of objects in kinematic pairs need to be considered. This introduces the need to automate the shape creation and modification decisions that take place during the design process. The innovative design paradigm is thus necessary for designing or altering kinematic pairs. In most existing Computer-Aided Design systems (CAD) [Encarnacao83], the decision on the creation of an object's shape is the task of the human designer; the CAD system is responsible for handling and verifying the consistency of the design decision. CAD systems that are capable of introducing shape variations to objects do so by modifying the

values of predefined shape parameters (routine design).

Having recognized the need for reasoning about the shape of objects and their relation to kinematic behavior, we developed a first principles theory based on the notion of configuration spaces. We have showed in previous work that *configuration spaces* can be used as an intermediate symbolic representation that relates object geometry and kinematic behavior ([Joskowicz87a]). The configuration space of a mechanism specifies the set of placements (positions and orientations) of objects in a mechanism such that no two objects overlap. It defines the set of legal motions for each object, and relates the functional description of a mechanism with the precise description of the shape of these objects. Partitioning the configuration space into regions provides us with a discrete, qualitative representation of the kinematic behavior. The following scheme, from [Joskowicz87c] illustrates these relationships:

Kinematic Behavior	
$\Updownarrow$	Regions of the configuration space
Configuration Space	
$\Updownarrow$	Contacts between objects
Object Geometry	

Given a description of the geometry and initial placement of each object in the mechanism, the analysis of the mechanism consists of deducing its kinematic behavior. Design, on the other hand, consists of taking a description of the desired kinematic behavior and producing object shapes that satisfy the requirements. Previous papers showed how to use this paradigm to analyze the kinematic behavior of a given mechanism. In this paper, we will show that this paradigm is also suitable for innovative design of kinematic pairs.

This paper is organized as follows: section 2 provides a precise statement of the design problem, together with a list of design constraints frequently encountered. A restricted language for kinematic behavior specification is also presented. Section 3 contains the main theory of shape design based on the manipulation of configuration spaces. We present a general backtracking algorithm for the design of kinematic pairs. Section 4 examines the case in which the two objects are convex. Section 5 discusses several efficient special-purpose algorithms for the case in which both objects have translational freedom only. Sections 6 and 7 show how produce configuration spaces when the specifications of the desired behavior are given in qualitative or partial terms. We conclude by comparing our work to previous work, and by discussing the limitations and possible extensions of our approach.

2 Statement of the Problem

Kinematic pairs are of great importance in mechanical design since they constitute the elementary building blocks of mechanical devices. In this paper, we deal with the design of composite kinematic pairs. Simple kinematic pairs, such as prismatic joints, screw-bolt pairs have a single kinematic behavior describable as a rotation or a translation (or a combination of both) around a fixed axis in space. Composite kinematic pairs. on the other hand can have several different kinematic behaviors along a number of axes.

In designing kinematic pairs, we distinguish between two variants of the design task: *new design* and *redesign*. In new design, for a given description of the desired kinematic behavior of the pair, the goal is to determine the precise shapes of the two objects that achieve this behavior. In redesign, in addition to the desired kinematic behavior, initial shapes for the two objects are also given. The goal is then to modify the given shapes of the objects to obtain the desired behavior. The theory presented in this paper deals with both variants uniformly.

As an example of a design task, consider the following design scenario: we are given a rotating disc A and a translating rectangle B as shown in Figure 2.1(a). Our design goal is to modify the shapes of the objects so that for two specific orientations of A, 0 and $\pi/2$, B prevents the rotation of A. For all other orientations, the motions of A and B must remain independent. A possible solution is to modify the shape of A by introducing two slots that allow B to create new contacts that prevent the rotation of A (see Figure 2.1(b)).

2.1 Design Spaces and Assumptions

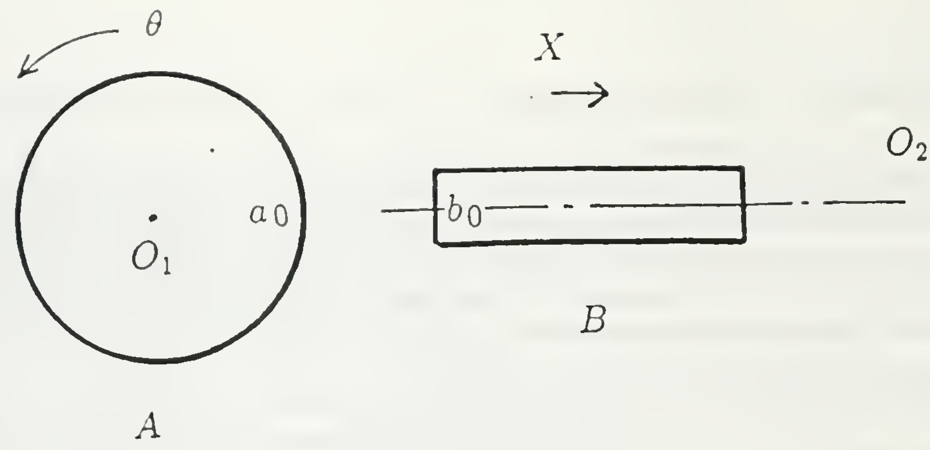
We begin by introducing several assumptions concerning the nature of the objects. In the following we assume that there are only two objects involved and:

1. Objects are two-dimensional.
2. Object contours consist of points (vertices), line segments (edges) and circular arcs.
3. Objects move on a plane, and each object moves along or around an axis that is fixed.

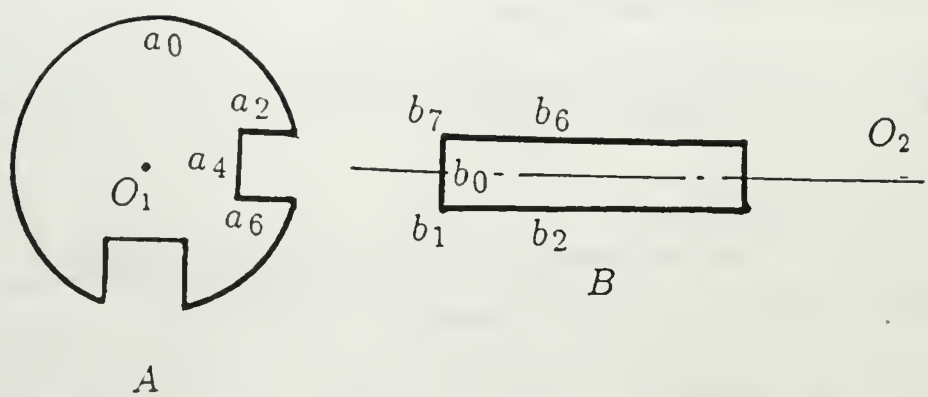
Since objects can only move on a plane, and along a fixed axis, their possible motions are restricted to rotations, translations, or no motion. Therefore, their interactions can be described with two-dimensional configuration spaces. We distinguish between five *design spaces*, corresponding to the degrees of freedom of each object in the pair: fixed-rotation, fixed-translation, translation-translation, rotation-translation and rotation-rotation.

2.2 Functional Specification of the Desired Behavior

The kinematic behavior of a mechanism can be described in terms of *possible motions* or in causal terms [Joskowicz87a]. Both descriptions are functional since they specify motion relationships between objects without referring to their actual geometry.



(a) Initial Shapes



(b) Modified Shapes

Figure 2.1: A Design Example.

A possible motions description specifies all the possible motions that each object (represented by a reference point) can have, together with the relationships between these motions. Every degree of freedom is associated with a motion parameter. The relationships between motions are specified by a function relating motion parameters. Functions can be real-valued or qualitative, indicating whether the motion parameters' ratio is increasing, decreasing or constant. Each motion parameter is bounded by intervals that define its legal range. Since we assumed that objects are two dimensional and move on fixed axes, an object A can only have one of the following three types of motions:

- A is fixed at point p: $fixed(A, p)$
- Possible rotation around axis O: $p\_rotation(A, O, \theta)$, $\theta \in [\theta_{min}, \theta_{max}]$
- Possible translation along axis O: $p\_translation(A, O, X)$, $X \in [X_{min}, X_{max}]$

Kinematic behavior can be described as the union of several possible motion *regions*. For example, all the reachable behaviors of the pair in Figure 2.1(b) are described as the union of three regions:

R_0: $p\_rotation(A, O_1, \theta)$, $p\_translation(B, O_2, X)$, for $\theta \in [0, 2\pi]_{mod 2\pi}$ and $X \in [X_0, \infty)$

R_1: $fixed(A, \theta)$, $p\_translation(B, O_2, X)$, for $\theta = 0$ and $X \in [X_1, X_0)$

R_2: $fixed(A, \theta)$, $p\_translation(B, O_2, X)$, for $\theta = \pi/2$ and $X \in [X_1, X_0)$

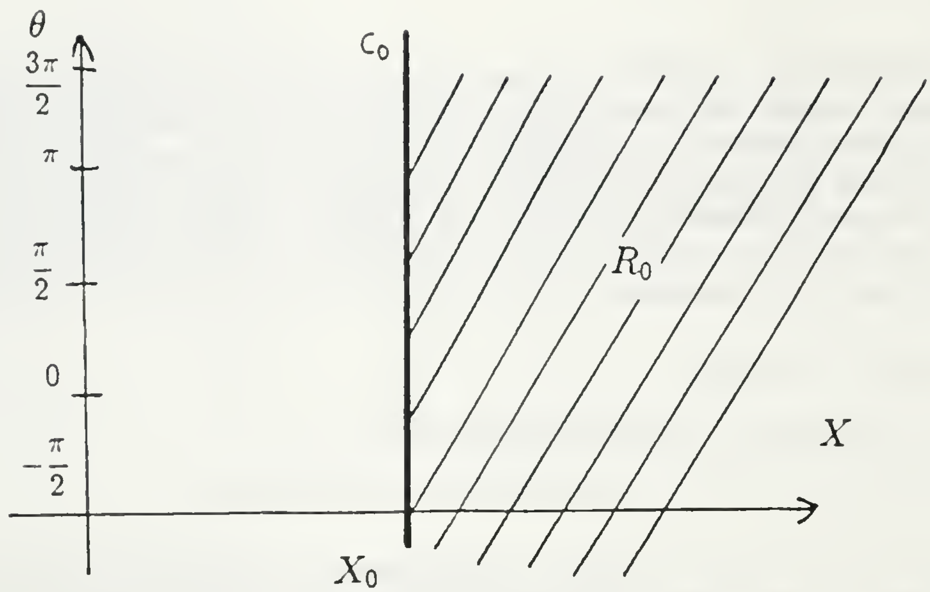
In R_0, the rotation of A and the translation of B are independent of each other since there is no function relating θ and X. In R_1 and R_2, A cannot rotate, but B can translate. Note that it is not possible to go directly from behavior R_1 to behavior R_2 without going first through R_0.

In a previous paper, we showed that there is a direct, one-to-one correspondence between possible motion descriptions and configuration spaces[1] [Joskowicz87a]. Relations between behaviors, described by possible motion labels, can be described by a a *region diagram*. A region diagram is a graph where each node represents a behavior and transitions between nodes indicate parameter conditions to be satisfied in order for a change of behavior to occur. Since each object has at most one degree of freedom, a two-dimensional configuration space fully describes the kinematic behavior of a pair of objects. Figure 2.2 shows the configuration space of the pair (A, B) before and after the modification. Note the direct correspondence between the above description and the regions of free object placements, indicated by hatched areas.

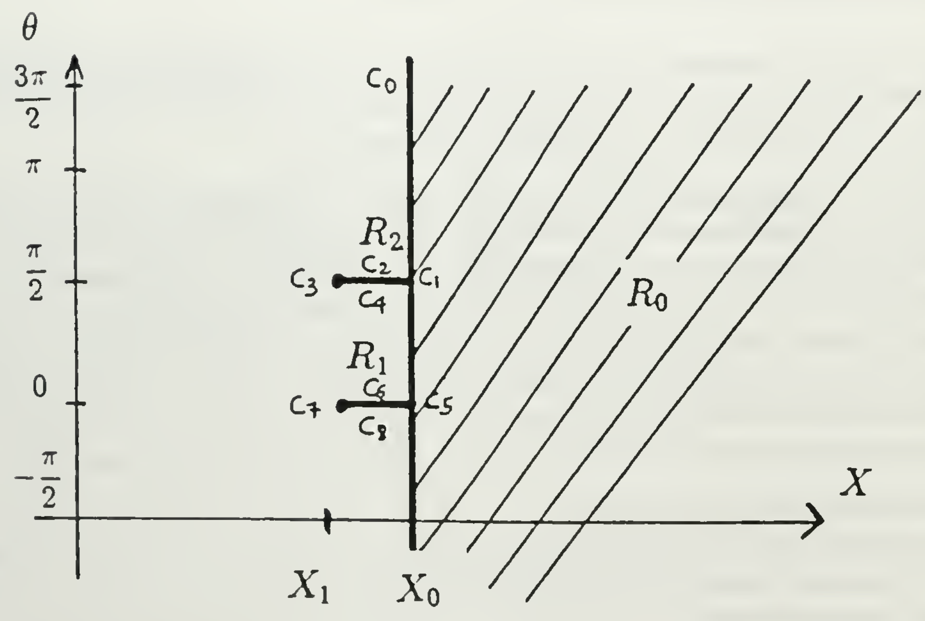
An alternative description of kinematic behavior is a *causal* description. This description states the effects that the motion of one object has upon the others (e.g., if A rotates clockwise then B rotates counterclokwise). The kinematic behavior of a mechanism can then be described by the motions of its objects resulting from a sequence of input motions. Section 7 shows that causal descriptions can also be mapped into an equivalent configuration space specifying the desired behavior.

Whether the description of the desired behavior is specified in terms of possible motions or in causal terms, the relationships between motions are sometimes specified qualitatively, rather than quantitatively. For example, we might require the clockwise rotation of object A to cause the translation of B to the right,

[1] The configuration space of a mechanism defines the set of *free placements* (position and orientations) of objects in a mechanism so that no two objects overlap [Lozano-Pérez83], [Schwartz83].



Initial Configuration Space $CO(A, B)$



Configuration space after the modification $R(A, B)$

Figure 2.2: The Configuration Spaces Before and After the Modification.

but the precise relationship between the two motion parameters is either unknown, or irrelevant. When the relationships between motions are described qualitatively, the goal is to find the exact shapes for the objects such that their actual kinematic behavior is qualitatively identical to the specified behavior. This extension will be discussed in section 6.

2.3 Design Constraints

In addition to kinematic requirements, design specifications contain other constraints that directly influence the final shape of the objects. Examples of such constraints are minimum object thickness, simplicity, and manufacturability.

Consider again the design example discussed at the beginning of this section, and the three alternative solutions presented in Figure 2.3. The first solution introduces a new part to B, used solely to fit into the slots of A. The second solution introduces new edges to B. The third solution is a drastic solution in which B is reduced to a single point, and the slots of A are of width zero. Note that all these solutions are kinematically equivalent, although none of them can be considered a good design solution. In the first solution, the portion added to B is too thin, making the mechanism unnecessarily fragile. The second solution introduced spurious lines and surfaces that do not play a kinematic role in the pair. The third solution is not physically feasible since a point is not an object, and slots of width zero cannot be manufactured. The solution shown in Figure 2.1(b) is the best design in this case since it is simple and easy to manufacture. Therefore, to guarantee a good solution, it is necessary to take into account *design constraints* in the shape design process.

We identify two types of design constraints: required constraints and optional constraints (see Table 1 for a summary of all design constraints). Required constraints provide a set of intervals for some important dimensions of objects, such as thickness, length of edges and arcs, angle between two edges etc. The most important of these constraints is the *physical feasibility* constraint. For two-dimensional objects, it requires objects to be topologically equivalent to a disk with a finitely many holes. It also rules out point objects.

Optional constraints put emphasis on other design aspects such as object simplicity and other non-kinematic aspects of the designed mechanism. For example, if A is to slide along B, we give preference to face-face contacts over face-vertex contacts since friction will cause the rapid deterioration of the contact point, changing the dimensions of one object as well as its behavior. Other constraints, such as not to allow the modification of a specific object, or part of it, come from design issues concerned with the role of the kinematic pair in the overall mechanism.

There are two basic methods of enforcing design constraints: a generate-and-test method and a direct method. In generate-and-test, a possible solution is first generated, and then validated against the design constraints. New solutions are produced until the validation succeeds. The direct method, on the other hand, takes into account the design constraints during the design so that the resulting design is always consistent with the constraints. Although most design systems use the generate-and-test approach, we found this approach to be impractical for the design of kinematic pairs. The reason is that purely kinematic requirements generally admit a very large number of solutions, most of them incompatible with the design constraints. The solutions we present in the next sections are all based on the direct approach.

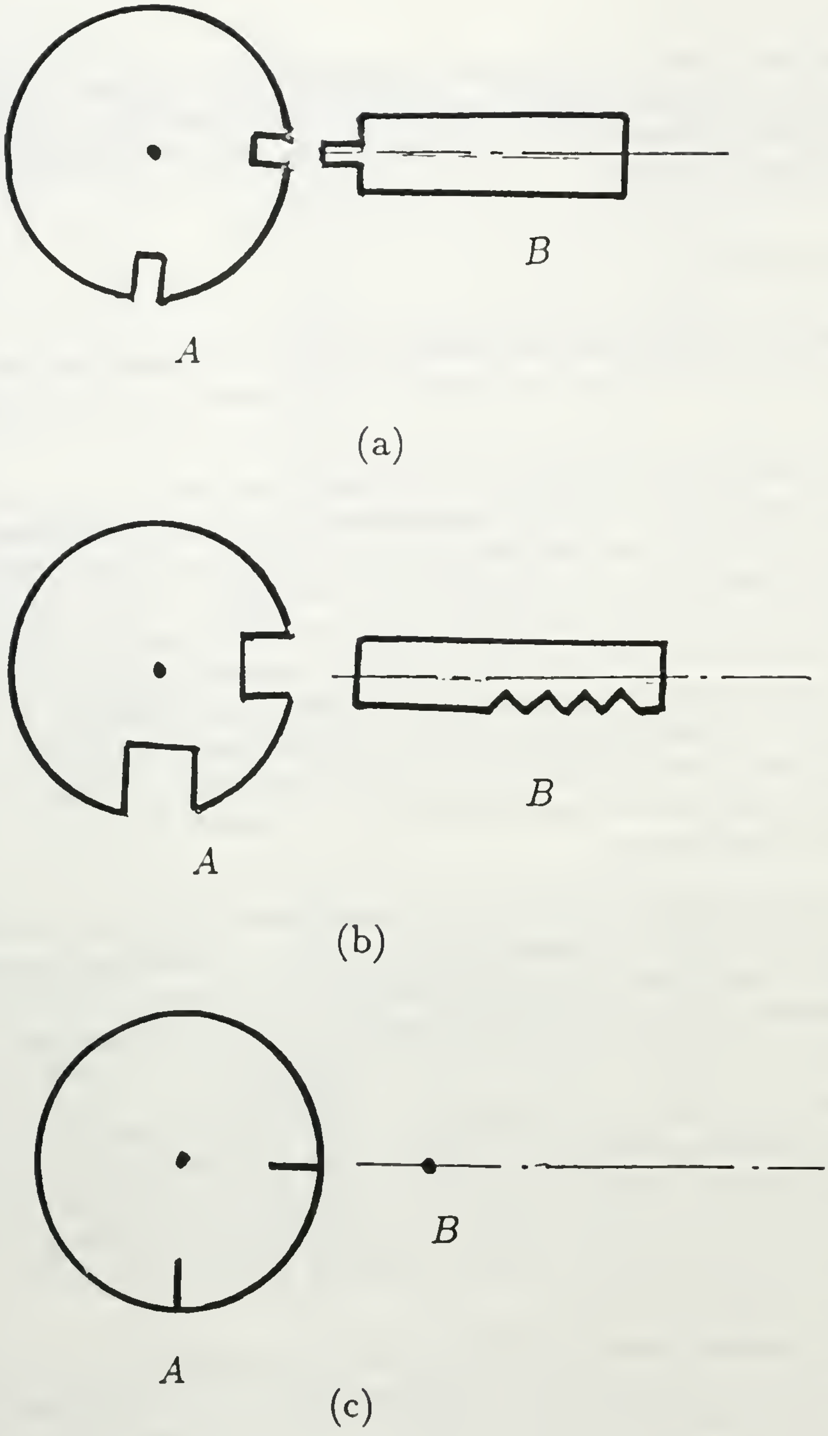
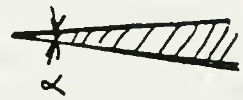


Figure 2.3: Three Examples of Poor Design Solutions.

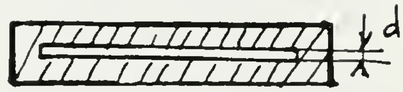
REQUIRED CONSTRAINTS

1. Physically feasible objects
2. Minimum thickness: $d \geq l_{min}$

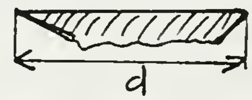
3. Minimum and maximum angles: $\alpha_{min} \leq \alpha \leq \alpha_{max}$



4. Minimum distance between lines: $d \geq l_{min}$



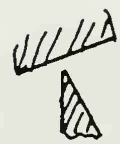
5. Minimum and maximum line lengths: $l_{min} \leq d \leq l_{max}$



6. Minimum and Maximum arc curvature: $r_{min} \leq d \leq r_{max}$

OPTIONAL CONSTRAINTS

1. Minimize the number of line-point contacts (prefer line-line contacts whenever possible).



2. Maximize the surface of line-line contacts

3. Use the minimum number of features (design the simplest objects)
4. Fix a bound on the complexity of one object (or both)
5. Allow only one object to be modified
6. Fix a bound on the complexity of one object (or both)
7. Allow modifications only in certain parts of the objects.
8. Allow modifications only of a certain type.

Table 1: Design Constraints

3 Shape Design from Configuration Space

We use configuration spaces as the basis of the design procedure. In this section, we assume that the desired pairwise behavior is given as a two-dimensional configuration space with exact boundaries.

Initially, we are given two objects, A, B (possibly empty), and a desired configuration space $R(A, B)$, corresponding to the desired kinematic behavior. The actual kinematic behavior of the objects corresponds to their actual configuration space, $CO(A, B)$. Comparing both the actual and desired behaviors amounts to comparing the two configuration spaces, $CO(A, B)$ and $R(A, B)$. The differences between them indicate where and how these behaviors differ. For example, in the previous design problem, the desired configuration space $R(A, B)$ contains two regions, R_1 and R_2, not present in $CO(A, B)$ (Figure 2.3).

The behavior of a kinematic pair can be modified by changing the boundaries of $CO(A, B)$ so that they match with the boundaries of $R(A, B)$. Boundaries of the configuration space are formed by the contact of two object features (a vertex, an edge, or an arc). Therefore, configuration space boundaries can be modified by removing contacts or introducing new ones. This in turn implies that the shape of the objects must be changed by adding and deleting edges and arcs to their contours. In the previous example, there are six configuration space boundaries, $c_2, c_3, c_4, c_6, c_7, c_8$, that must be added to $CO(A, B)$, and two that must be deleted (c_1 and c_5) to allow transitions from R_0 to R_1 and R_2 [2]. The design problem consists in finding a sequence of feature additions and deletions to the objects' contours so that the actual and the desired configuration space boundaries match and the design constraints are satisfied.

3.1 Boundaries of the Configuration Space

A boundary in the configuration space corresponds to the contact between the feature of one object and a feature of the other object. Boundaries separate regions regions of free placements and regions of forbidden placements. The form of the configuration space boundaries is determined by the design space and by the features that come in contact to create it. For example, in the rotation-translation space (one object rotates, the other translates), a vertex-edge contact produces a configuration space boundary with the following equation

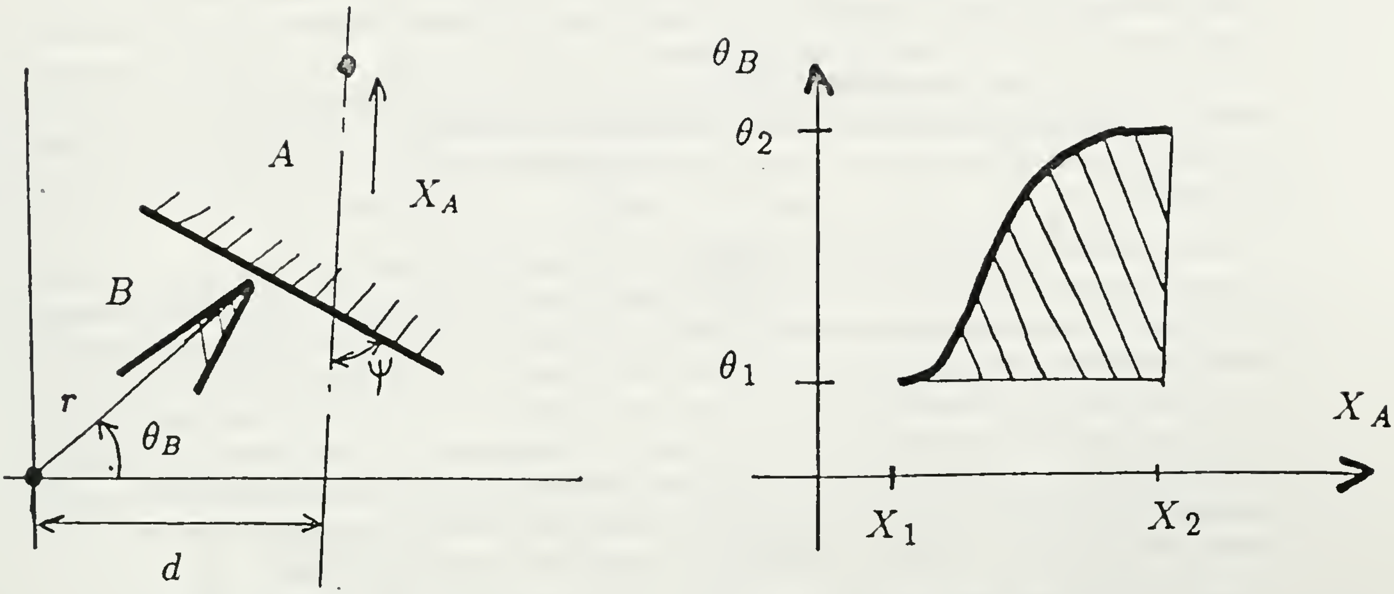
$$X_A = r[\sin\theta_B + \cos\theta_B \tan\psi] - d\tan\psi \tag{1}$$

where r is the distance from the rotating vertex to the rotation point of B, ψ is the angle of the edge of A with the translation axis, d is the nominal displacement of the rotation axis with respect to the translation axis (see Figure 3.1). Arc-vertex or arc-edge contacts produce (when the center of the arc coincides with the center of rotation) a configuration space boundary that is a line, such as the boundary c_0 in Figure 2.3 (a) produced by the contact (a_0, b_0).

$$X_A = X_0, \quad \theta_B \in [\theta_1, \theta_2] \tag{2}$$

We have classified the different types of boundaries that arise from the nine possible pairwise contacts in each of the five design spaces. The result is a table of elementary contacts that specifies, for each type of contact and design space, the type configuration space boundary produced, together with the

[2] Regions R_1 and R_2 are rectangles of width zero, and thus have four sides, two of which of zero length.



Edge-Vertex Contact

Corresponding configuration space boundary.

Figure 3.1: The Configuration Space Boundary Produced by an Edge-Vertex Contact in the Rotation-Translation Space.

set of equations that define it (see Figure 3.2). In this table, the type of configuration space boundary, together with the set of equations that define it are stored the slots of the table. The cases denoted by a prime (i.e., 10') indicate cases that are completely symmetrical to cases without a prime (i.e., 10). The spaces Fixed-Translation, Fixed-Rotation, and Rotation-Translation are symmetrical to Translation-Fixed, Rotation-Fixed and Translation-Rotation respectively. The table of elementary interactions is used to determine, given a configuration space boundary, what pair of features will produce this boundary. Once the possible types of contacts have been determined, the actual coordinates of the features must be determined. Appendix A provides a complete description of the elementary contacts for each design space together with the relations between the parameters of the configuration space boundary and the features that create it.

Given a desired configuration space boundary, the design task consists in finding a pair of object features that, when in contact, will create this boundary. Note that not every contact between features can produce a desired configuration space boundary. For example, in the rotation-translation space, a vertex-edge contact can never be used to produce a line boundary in $CO(A, B)$, since for no values of r, d and ψ, equation (1) represents a line. In this case, only a vertex-arc or an edge-arc contact can produce the desired boundary. This means that arc a_0 cannot be substituted by a vertex and still produce the boundary c_0 when in contact with b_0. Thus, the type of the configuration space boundary can be used to determine which pair of features can, in principle, produce the boundary. Having determined the type of contact, we then find the precise coordinates of the features that create the boundary.

Once we have identified the boundaries of the actual configuration space $CO(A, B)$ that have to be added or deleted, and determined what type of features can produce them (using the elementary interactions table), we still have the problem of deciding what alterations to make, and where. For example, suppose we need to delete a particular boundary of the configuration space. What were the features of A and B that produced it? It is not necessary to delete both the feature of A and the feature of B to delete the boundary. Which deletion will not alter the rest of the configuration space? When adding a feature to an object, where should we add it so as to keep the connectivity of the object? In order to answer these and other questions, a data structure relating boundaries of the configuration space to the object features that produced it is necessary. We call this data structure a configuration space boundary map.

3.2 The Configuration Space Boundary Map

Before turning to the problem of how to decide where to add or delete object features, we introduce a data structure that will be of much use in determining the effects of the modifications on the configuration space.

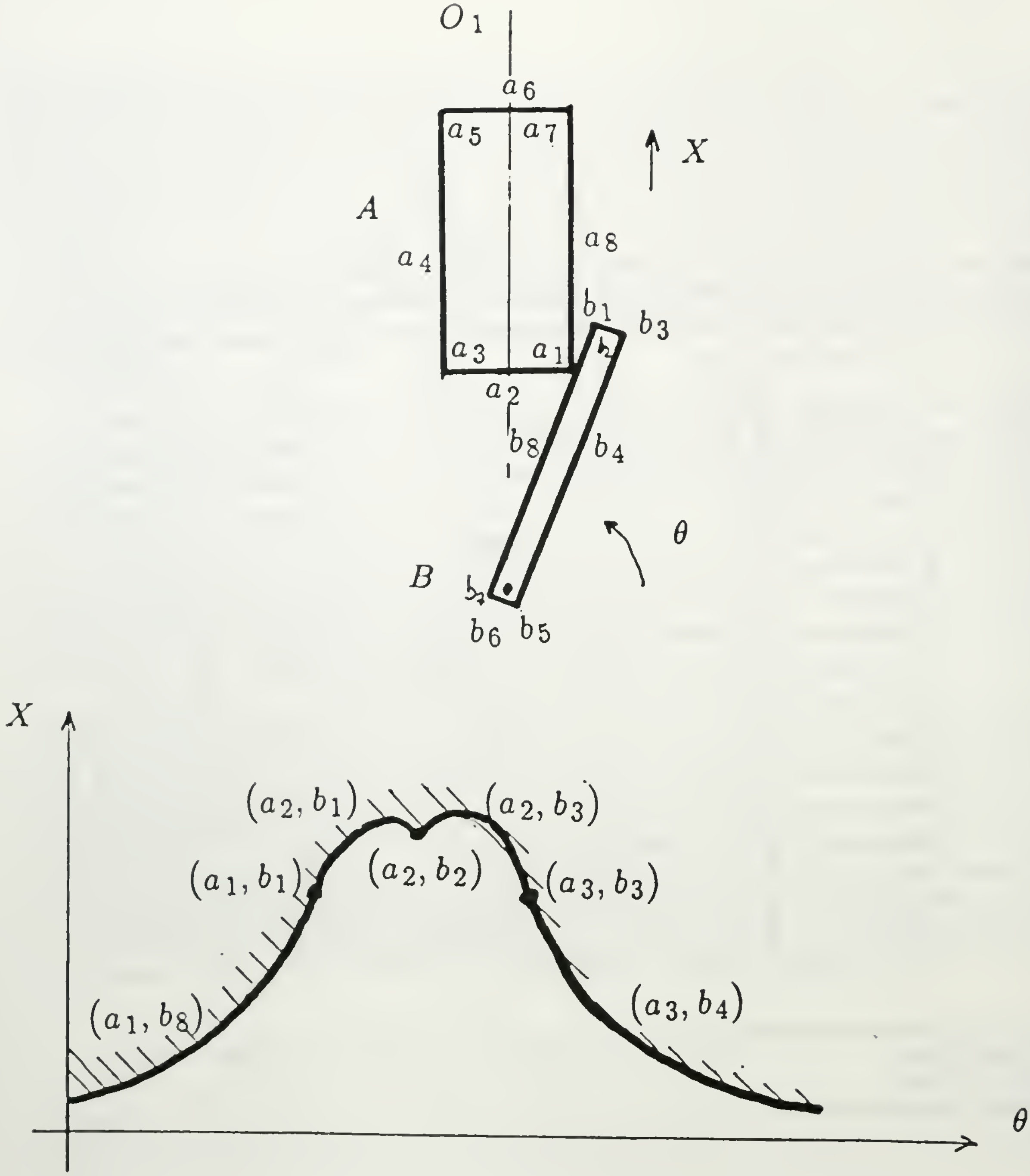
To every boundary of the configuration space we assign the pair of features, one from A and one from B that produced it (if several pairs produced the same boundary, we record all the pairs). We then keep these pairs in a linked list that describes the configuration space boundary in a clockwise order. If the configuration space has several disconnected regions, we keep one such list per region. See Figure 3.3 for an example of a configuration space map. Using this map, for a given configuration space boundary we can determine:

Design Spaces

Contact	Trans-Fixed	Rot-Fixed	Trans-Trans	Trans-Rot	Rot-Rot
vertex-vertex	Type 0	Type 0	Type 0	Type 0	Type 0
vertex-edge	Type 0	Type 0	Type 1	Type 3	Type 10
edge-vertex	"	"	"	Type 4	Type 10'
edge-edge	"	"	"	Type 0	Type 0
vertex-arc	"	"	Type 2	Type 5	Type 11
arc-vertex	"	"	"	Type 6	Type 11'
edge-arc	"	"	Type 1	Type 7	Type 12
arc-edge	"	"	"	Type 8	Type 12'
arc-arc	"	"	Type 2	Type 9	Type 13

Configuration Space Boundaries:
Type 0: Point
Type 1: Line
Type 2: Circular Arc
etc,

Figure 3.2: Table of Elementary Interactions.



$$-(a_1, b_8)-(a_1, b_1)-(a_2, b_1)-(a_2, b_2)-(a_2, b_3)-(a_3, b_3)-(a_3, b_4)-$$

Figure 3.3: Two Objects, their Configuration Space and a Portion of the Configuration Space Boundary Map.

- What pair of features produced it.
- Whether the removal of one feature will affect the other configuration space boundaries.
- What features are on the vicinity of a certain configuration space boundary, so that additions to the object boundary are connected to this boundary.
- How is the configuration space altered by the removal or addition of a feature.

The first query can be answered directly since every boundary has associated with it the pair of features that produced it. The second query can be answered by scanning the linked list, and looking for a pair that contains the desired feature. If no such pair exists, then the removal of the feature will not alter the configuration space. For example in Figure 3.3 the removal of the edge a_6 belonging to A will not affect the configuration space $CO(A, B)$ since this edge does not appear in any pair of the list and thus does not influence it. The vicinity query can be answered by looking at the neighbors in the list of the pair corresponding to the boundary. Finally, when deleting a feature, all the boundaries for which the feature appears in the pair must be revised.

Features producing boundaries that appear in the configuration space map are said to be *functional* because their alteration changes the configuration space. Non-functional features are features that do not come in contact with other features and thus play no role in the definition of the configuration space. They are important, however, to keep the objects' contours closed.

3.3 A Backtracking Algorithm for Shape Design

The design procedure starts by comparing the actual and the desired configuration spaces. The goal is to delete the configuration space boundaries of $CO(A, B)$ that do not match boundaries of $R(A, B)$ and to add to $CO(A, B)$ the boundaries that appear in $R(A, B)$ but not in $CO(A, B)$. Two boundaries match iff their form is identical and the free object placements lie on the same region.

For each boundary difference, a pair of object features to either delete or add the required boundary is selected. For a deletion, at least one of the features that contributed to the boundary creation must be deleted. For an addition, one or two new features must be created to produce the boundary. The type of features that produce the boundary in question is determined from the table of elementary contacts. For example, in order to delete c_1, either a_0 or b_0 must be deleted. In order to add c_2, it is sufficient to add the edge a_2 (but not an arc) since its contact with edge b_6 creates c_2.

In both cases of addition and deletion, there might be more than one candidate feature pair and thus a (nondeterministic) choice must be made. For example, c_3 can be created with the existing edge b_0 and a new edge a_4, or with a new arc b_9 and a new edge a_4. In this case, the first choice is preferred since it introduces fewer new features. After every object contour change, the configuration space $CO(A, B)$ is updated. If the new features violate a design constraint (except closed contour), the pair is rejected and a new candidate pair is selected. This guarantees that a bad choice is rejected as soon as a violation occurs, instead of waiting until the whole design process is completed. Note that the final designed objects might not be consistent, i.e., their contour might not be closed. For example, if we remove the edge b_0 from B, and take A as shown in Figure 1b, we still have that $CO(A, B) = R(A, B)$, although B does not have a

closed contour An attempt to "fill in" the missing contours is made, without altering $CO(A, B)$. If this attempt fails, the algorithm backtracks over its previous choice. The design process is successful when all the differences between $CO(A, B)$ and $R(A, B)$ have been eliminated, and both objects are consistent with the design constraints. Here is a design-space independent backtracking algorithm.

Procedure DESIGN($A, B, R(A, B)$, Design-Constraints)

1. Compute $CO(A, B)$
2. DELETE := boundaries in $CO(A, B)$ that do not match boundaries in $R(A, B)$
   ADD := boundaries in $R(A, B)$ that do not match boundaries in $CO(A, B)$.
3. While $CO(A, B) \neq R(A, B)$ do
   3.1 For a boundary b_i in ADD, do
      a Using the table of elementary interactions, determine the type of features that can produce the type of boundary of b_i.
      b Choose a pair of features (a, b) of the appropriate type that produce b_i. Prefer pairs in which one of the features is already existing, and is connected to the object boundary.
      c Check whether the new feature(s) comply with the design constraints.
   3.2 Update $CO(A, B)$, ADD and DELETE.
   3.2 For a boundary b_i in DELETE, choose a feature from the pair that created it and delete it from the corresponding object. Do not delete new features.
   3.3 Update $CO(A, B)$, ADD and DELETE.
4. End
5. Complete the object contours without modifying $CO(A, B)$.
   If this is not possible, return "FAIL".

The analysis of feature contacts (section 5.1 and Appendix A) reveals that the equations relating a configuration space boundary c_i to the features that created it are underconstrained when only c_i is given. Thus, there is, in principle, an infinite number of coordinate values for features to create a new configuration space boundary, leading to an infinite number of feature choices. Nevertheless, for most of the interesting design cases, the number of choices is finite. When one of the objects (B) is not allowed to change, the number of possible choices of features of B that can participate in the creation of a new boundary is bounded by the number of features in B. Also, if only one new object feature is introduced at a time (to either A or B, but not both), the number of choices is bounded by the number of features of A and B. In this case, we can impose additional constraints, such as the continuity of the boundaries of both A and B, to completely determine the choice.

From the previous considerations it follows that when the number of choices is finite, the complexity of the algorithm is exponential in the number of choices. In order to improve the real run time of the algorithm, we introduce two heuristics: Boundary Ordering and Feature Ordering. Both heuristics are

based on the following principle of local convexity.

Principle of Local Convexity:

Consider again the configuration space boundary map in Figure 3.3. Note that most adjacent boundaries in the configuration space boundary map share one common feature of either A or B. Moreover, when a feature changes, the change is generally to an adjacent feature in the objects. This is because the change of contact is always to an adjacent feature when two objects are convex. However, in the case of concave objects, "jumps" occur from time to time. Thus, when selecting a pair of candidates to produce a configuration space boundary, it is a good idea to examine first the features that are immediately adjacent to the ones that produced the previous configuration space boundary. The next best choice are features that are at a distance of one feature from it, and so on.

The Boundary Ordering heuristic determines the order in which the boundaries in ADD and DELETE are examined. The ordering that follows the principle of local convexity is a clockwise (or counter clockwise) ordering of the boundary differences between $CO(A,B)$ and $R(A,B)$. The Feature Ordering heuristic determines, for a given configuration space boundary, the order in which potential candidate features of both objects will be examined. Following the principle of local convexity, the best candidate features are the ones immediately adjacent to the ones that produced the contiguous configuration space boundaries. The next best choice are features that are at a distance of one feature in the clockwise direction, then the ones at a distance of two and so on. As we will see in the next section, these heuristics yield linear time algorithms for the case of two convex objects.

The main advantage of this backtracking algorithm is that it is design space independent: it can be applied indistinguishably to the 5 design spaces. However, its complexity is exponential on the number of features of both objects. By examining each of the design spaces separately, it is possible to find more efficient design algorithms. The next two sections discuss in detail the design of two convex objects, and design in general in the translation-translation space. In both cases, efficient design algorithms are presented. Further research is necessary to develop special purpose algorithms for the case of non-convex objects in spaces that involve a rotation.

4 Design of Convex Objects

In this section, we present two theorems for the case of convex objects that yield to efficient design algorithms and provide further evidence for the use of the Boundary Ordering and Feature Ordering heuristics.

Let $A = \{a_i\}$ and $B = \{b_j\}$ be two simple polygons defined by their contour features a_i and b_j. Let n and m be the number of features of A and B, respectively. Let

$$boundary(CO(A,B)) = \{C_i \mid C_i = \{c_j^i = < a_j, b_j >\}\}$$

be the set of connected components C_i of the boundary of $CO(A,B)$, and let $\{c_j^i\}$ be the set of contour features of component C_i. The boundary feature c_j^i was produced by the contour features a_j of object A and b_j of object B. Let the predicate $adjacent(x_i, x_j)$ be true when the feature x_i is adjacent to x_j.

Convexity Theorem: General Case

Let A and B be two convex objects, and let $CO(A,B)$ be their two-dimensional configuration space. The following two properties hold:

1. Two adjacent configuration space features in $CO(A,B)$ are produced by a feature of A and two adjacent features of B, or by a feature of B and two adjacent features of A. In addition, at least one of these features is a vertex:
   $\forall c_i^k, c_j^k \in C_k,\;\; c_i^k = < a_i, b_i >,\;\; c_j^k = < a_j, b_j >$
   $adjacent(c_i^k, c_j^k) \iff [a_i = a_j \wedge adjacent(b_i, b_j)] \vee [b_i = b_j \wedge adjacent(a_i, a_j)]$
   and at least one of the features a_i, a_j, b_i, b_j is a vertex.

2. The upper bound of the size of $CO(A,B)$ is $O(nm)$.

For the translation-translation space, this theorem has a stronger version:

Convexity Theorem: Translation-Translation Case

In addition to the conditions of the previous theorem, let $CO(A,B)$ be a translation-translation space. Let the contours of A, B, $CO(A,B)$ be ordered in counterclockwise, clockwise, and counterclockwise order, respectively. Then the following two properties hold:

1. The boundary of $CO(A,B)$ is a simple convex polygon of size $O(n+m)$.

2. Configuration space boundaries are produced by the alternation of a feature of A and a feature of B, in increasing order:
   $\forall\, c_i, c_{i+1}, c_{i+2}, c_{i+3} \in\; boundary(CO(A,B))$,
   $c_i = < a_i, b_i >,\; c_{i+1} = < a_{i+1}, b_i >,\; c_{i+2} = < a_{i+1}, b_{i+1} >,\; c_{i+3} = < a_{i+2}, b_{i+1} >$

   (indices are modulo k, where k is the number of features in $CO(A,B)$.)

Before we turn to the proof of these theorems, let us consider their consequences.

The first direct consequence of the Convexity Theorem is that contact transitions between object features have a specific structure. For example, it is not possible to go directly from a vertex-vertex contact to an edge-edge contact since this would violate the adjacency requirement. Figure 4.1 shows the possible contact transitions in the form of a graph structure. In this graph, each node represents a type of contact and a link between two nodes represents a possible transition between contacts. Note that all contact transitions involve at least one vertex.

The Convexity Theorems have important implications with respect to the backtracking algorithm for shape design presented in section 3.3. Because of the additional adjacency constraints imposed by the theorems, not every object feature is a viable candidate to produce a desired configuration space boundary. Also, the adjacency requirement suggest that the sequential traversal of the configuration space boundary leads to a sequential construction of the object boundaries. These two remarks are captured by the following two properties. Assume (wlog) that we traverse each connected component of the configuration space in clockwise order. Then:

Property 1: The only viable candidate features to produce the next configuration space boundary are features that are adjacent to the ones that produced the previous configuration space boundary.

Property 2: The contours of the designed objects must be connected as they are designed. This means that feature additions that are not connected to the existing contour are not to be considered. In addition, the contours of the designed object(s) can be closed by adding an edge between the two endpoints of the open boundary.

The consequence of the first property is that at every choice point in the backtracking algorithm, only four candidate feature pairs are possible. Let $c_i = < a_i, b_i >$ be the boundary of $CO(A, B)$ just examined. Then, to produce the next boundary c_{i+1} there are only four possible pairs of features:

1. $c_{i+1} = < a_{i-1}, b_i >$, where a_{i-i} is the feature proceeding a_i in the contour of A, and already present. In this case no new feature is added; two existing features are re-used.
2. $c_{i+1} = < a_i, b_{i-1} >$, where b_{i-1} is the feature that proceeds b_i in the contour of B, and already present. In this case no new feature is added; two existing features are re-used.
3. $c_{i+1} = < a_{i+1}, b_i >$, where a_{i+1} is the feature following a_i. This is a new feature whose endpoint coincides with the endpoint of a_i.
4. $c_{i+1} = < a_i, b_{i+1} >$, where b_{i+1} is the feature following a_i. This is a new feature, whose endpoint coincides with the endpoint of b_i.

Therefore, even when the contours of both objects are not specified, the number of choices to produce the next configuration space boundary is reduced from infinity to four (see the discussion in section 3.3). The extra constraint on the adjacency of the new feature determines completely the system of equations defining the endpoints of the new feature (see section 5.1 and Appendix A).

In addition, the close consideration of the four possible feature choices reveals that the choice between the four alternatives can be made by a local computation in constant time. Consider the three spaces that

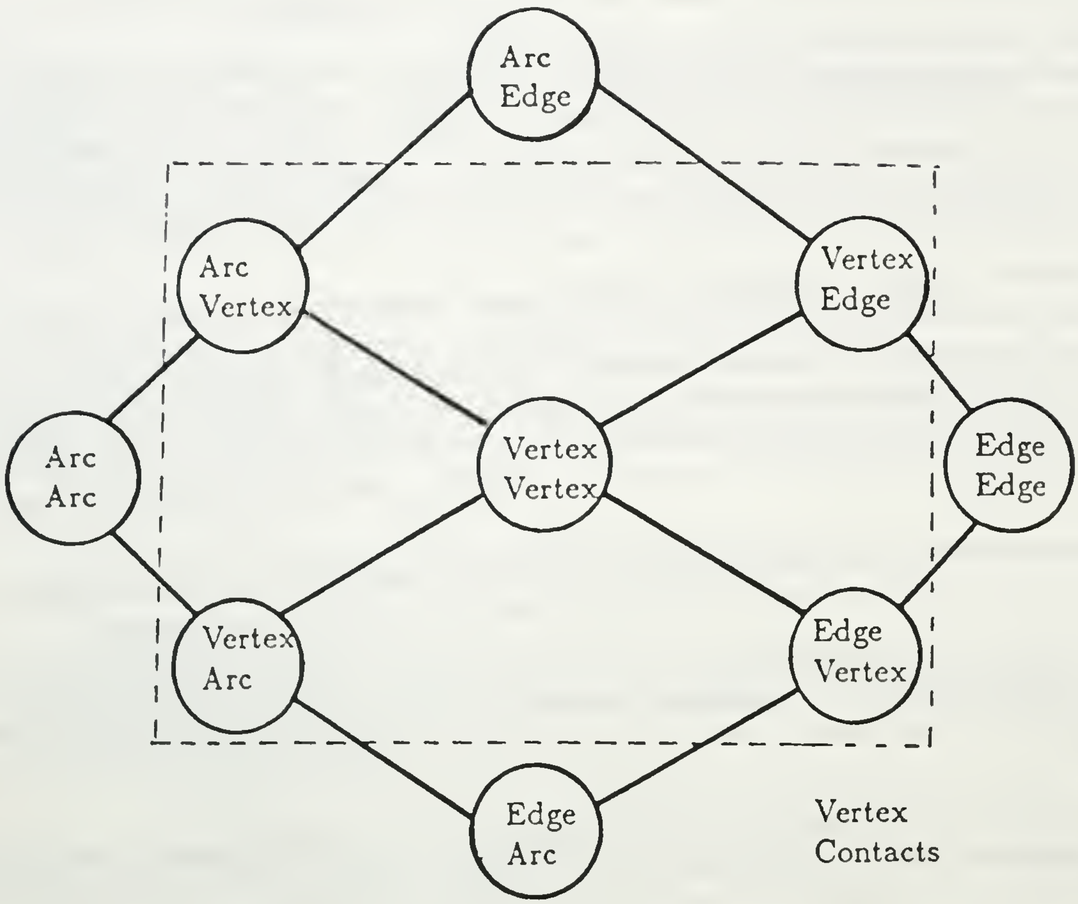


Figure 4.1: Contact Transition Graph for Convex Objects.

include a rotational degree of freedom: Rotation-Translation, Translation-Rotation and Rotation-Rotation. By the table of elementary interactions in Figure 3.2, we see that each contact produces a different type of configuration space boundary. From the contact transition graph, we can conclude that contact transitions necessarily imply a change in the type of contact. Therefore, the configuration space boundary uniquely determines the next type of contact. For example, suppose that we are designing two objects in the Translation-Rotation space. Suppose that the boundary $c_1 = < a_1, b_1 >$ was produced by a vertex-vertex contact, and that the next boundary $c_2 = < a_2, b_2 >$ is of Type 3. Then, the only possible contact that can produce it is a vertex-edge contact. This implies that the vertex a_1 that was used to produce c_1 is used again ($a_1 = a_2$), and that b_2 is an edge whose endpoint is b_1. From this adjacency requirement, we can determine exactly the precise form of the edge b_2. Thus, assuming that no special cases occur, the choices are reduced to a local computation carried in constant time. Special cases occur when, for example, for specific parameter values a boundary of one type becomes a boundary of another type - as in the case of an arc having infinite radius and that becomes a line segment. These cases must be handled separately.

For the Translation-Translation space, the only case in which there are two possible choices with the same type of boundary are the edge-edge contact (transition to edge-vertex and vertex-edge, both of type 1) and the arc-arc contact (transition to arc-vertex and vertex-arc, both of type 2). However, the Convexity Theorem for the Translation-Translation space rules out one of the possibilities since it requires the alternation of vertex contacts between A and B.

From the previous discussion we conclude that it is possible to design either one or both object contours in time linearly proportional to the size of the boundary of the desired configuration space. This presupposes that we can find the correct starting features of A and B such that $c_1 = < a_1, b_1 >$, where c_1 is the first boundary feature in the clockwise ordering of $boundary(CO(A, B))$. If we assume that c_1 is a boundary of type 0 (a point) produced by a vertex-vertex contact, then the precise coordinates of the two starting vertices a_1 and b_1 can be found. These observations are contained in the following two theorems:

Theorem 1:

Let $R(A, B)$ be a two-dimensional configuration of size k that can be produced by two convex objects. Then it is possible to design two objects, A and B, such that $CO(A, B) = R(A, B)$ in time proportional to k. In addition, $k = O(nm)$, where n and m are the number of features of A and B, respectively.

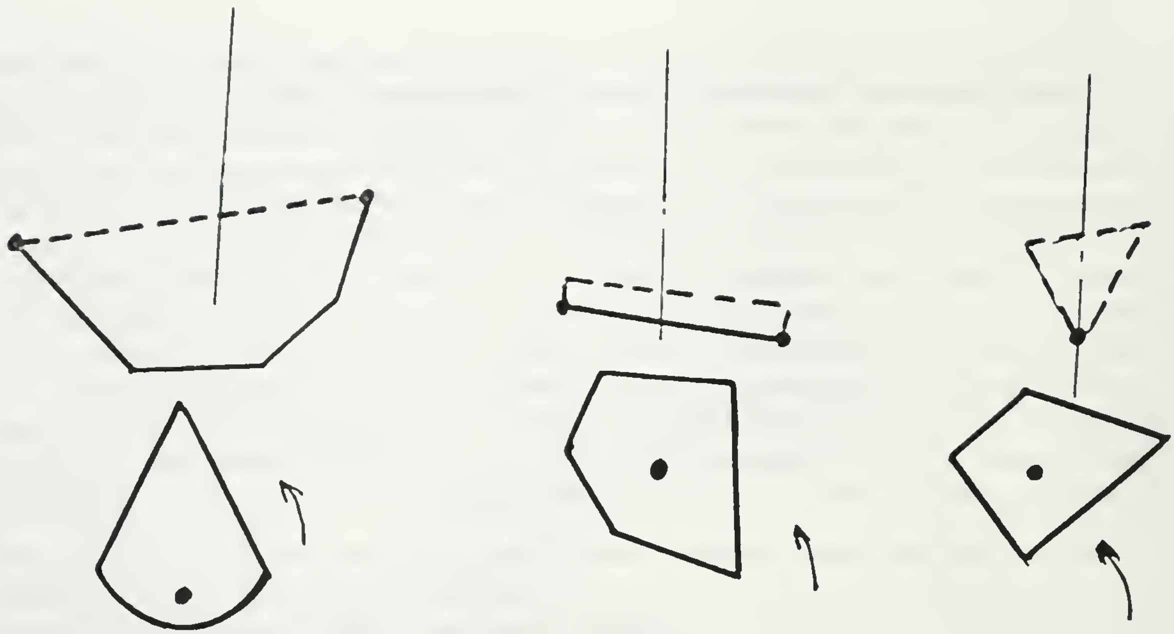
Algorithm 4.1:

The design procedure starts by determining the starting vertices of A and B from starting point in $boundary(CO(A, B))$. The design proceeds by following in clockwise order the boundary of $CO(A, B)$ and producing the boundaries of A and B in clockwise and counterclockwise order, respectively. The last step consists of closing the contours of A and B, if this is necessary. There are three cases when a final boundary modification is necessary (see Figure 4.2):

1. The contour consists of more than two features that are not vertices. The contour is continuous, and an edge between its starting and ending endpoints closes the object without altering the configuration space.

2. The contour is a single edge. Thickness is added to make this edge a real object.

3. The contour is a single vertex. In this case, the object can be constructed by adding two edges that



(a) (b) (c)

Figure 4.2: Three Cases of Object Completion.

meet at the vertex.

If $CO(A, B) \neq R(A, B)$, then the desired configuration space cannot be produced by two convex objects.

Proof of Theorem 1: By the Convexity Theorem, Properties 1 and 2, and the previous considerations, it follows that the only choices that can produce the configuration space boundaries will be appropriately examined. Furthermore, a local computation can determine the correct choice.

Theorem 2:

Let $R(A, B)$ be a two-dimensional, Translation-Translation configuration space of size k. Then it is possible to design two objects, A and B, such that $CO(A, B) = R(A, B)$ in time proportional to k. In addition, $k = O(n + m)$.

The proof makes use of the Convexity Theorem for the Translation-Translation space and is symmetrical to the proof of Theorem 1.

Proof of the Convexity Theorem

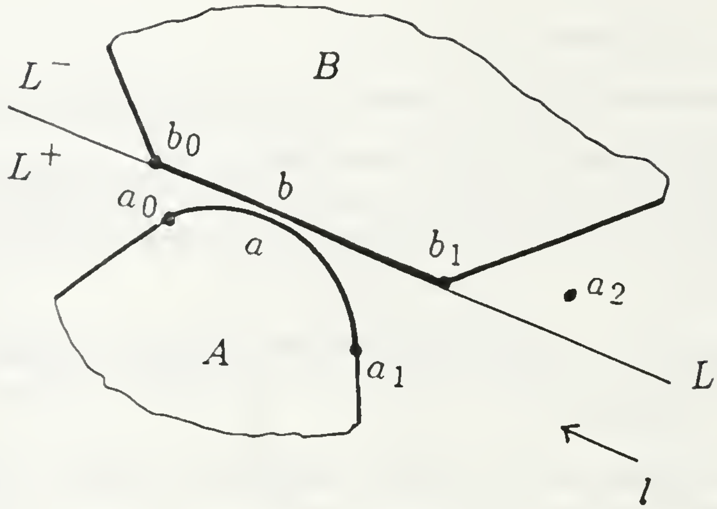
The theorem is proven by examining, for each of the design spaces, every contact and the contacts that immediately follow it. Each case is proven by arguing that any contact different than the one specified by the adjacency requirement in the Theorem violates the convexity assumption.

In proving each case, we use the following construction. Let A and B be two convex objects in contact, and let a and b be the two features in contact. Let a_0 and a_1 be the features immediately adjacent to a and let b_0 and b_1 be the features immediately adjacent to b, as shown in Figure 4.3. Let L be the line that is tangent to the point of contact (in the case of an edge-edge contact, the line is parallel to both edges; in the case of a vertex-vertex contact, the line lies anywhere between a_0 and b_0 or a_1 and b_1). Let $\mathbf{l}$ be the vector that generates L. Since A and B are both convex, the line L completely separates both objects. Let L^+ be the side of A and L^- the side of B. For each case, we prove that if a contact other than (a, b_1), (a, b_0), (b, a_0) or (b, a_1) immediately follows the contact (a, b), then one of the features that produces it lies on the "forbidden" side of the line (for a feature of A on the L^- side, for a feature of B on the L^+ side of the line). Having a feature in the forbidden side of the line directly implies that the object is not convex.

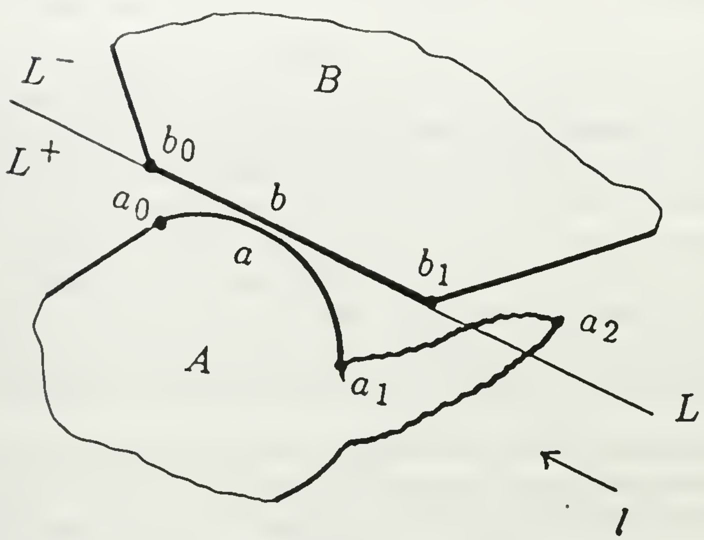
The proof of each case is done by contradiction. We begin by assuming that a non-adjacent feature, say a_2 or b_2, appears in the contact following the current (a, b) contact. We then show that this feature must lie on the forbidden side of line L, and thus violates the convexity assumption.

Translation-Translation Space: Suppose A is fixed and B has 2 degrees of translational freedom (this is equivalent to having A and B with one degree of translational freedom each). Let B move in the direction of $\mathbf{l}$ (see Figure 4.4a). In this case, the next contact is either (a, b_0) or (a_0, b). Suppose, by contradiction, that it is not. Then, one of the features must lie on the forbidden side of the line so that the contact occurs *before* the two contacts specified above. This violates the convexity constraint. The case in which B moves in the direction opposite to $\mathbf{l}$ is symmetrical.

Translation-Rotation Space: Suppose A rotates and B translates. Let A turn in a clockwise direction (see Figure 4.4b). Then four types of contacts can occur: $(a_1, b), (a, b_0), (a, b_1), (a_0, b)$. Take, for example, (a_1, b). Suppose that the contact (a_2, b) takes place instead. Then, if we position A and B so that (a_1, b) is in contact, we find that a_2 necessarily lies on the L^- region and a_2 appears *before* a_1 in the



(a) Canonical Object Contact.



(b) Violation of Convexity.

Figure 4.3: Canonical Contact of Two Convex Objects.

counter-clockwise ordering. This implies that the boundary of A crosses the line L, violating the convexity assumption. The other three cases can be proven in a similar way.

Rotation-Rotation Space

Exactly along the lines of the previous case. See the four possible contacts in Figure 4.4.

Since every feature of A can come in contact with every feature of B, the maximum size of $CO(A, B)$ is $O(nm)$.

Proof of the Convexity Theorem, Translation-Translation Case

This theorem, including the linear bound on $CO(A, B)$, has been proven by Lozano-Pérez [Lozano-Pérez83] (see also section 5.1).

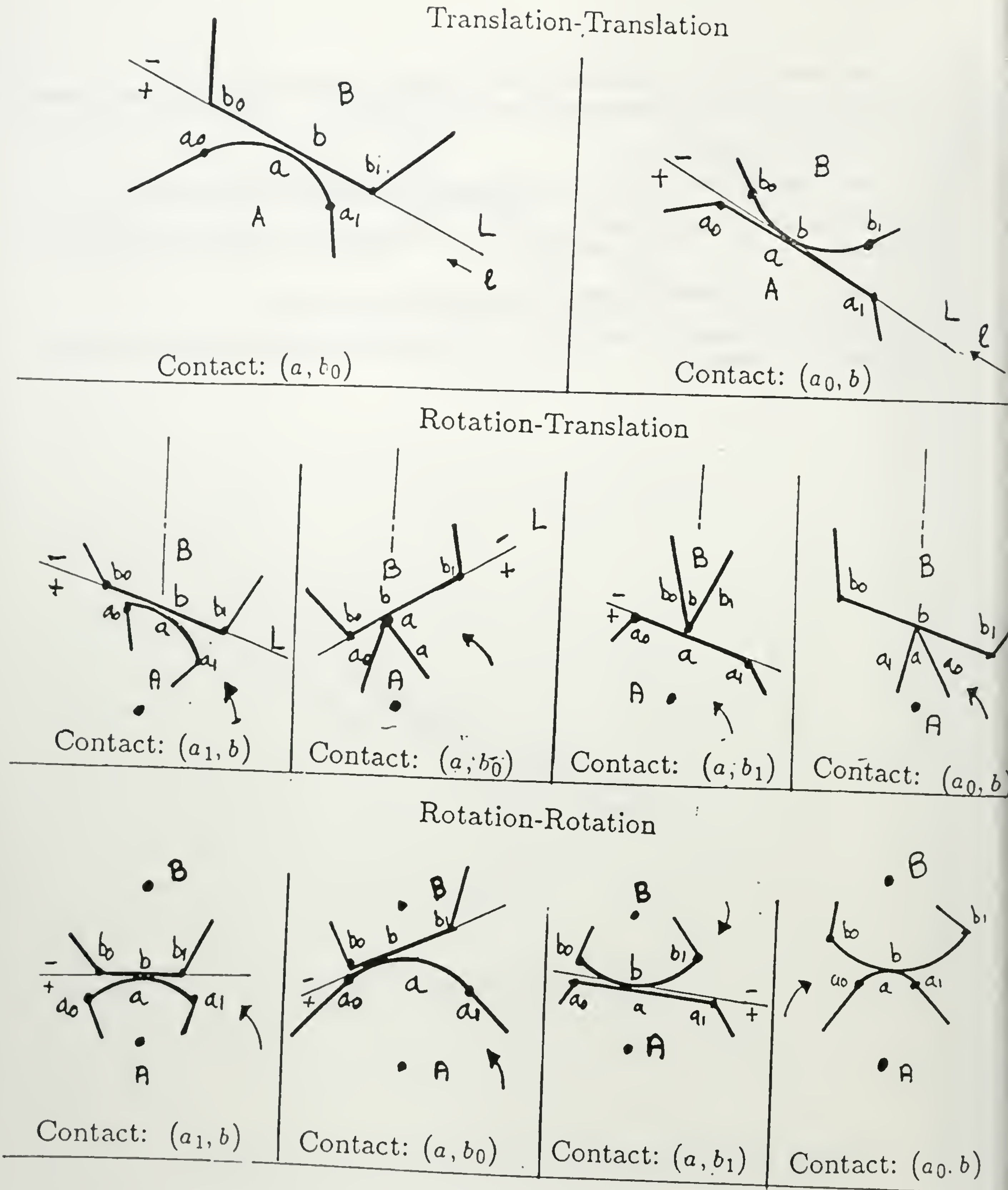


Figure 4.4: Case Analysis for Convexity Theorem.

5 Design in the Translation-Translation Space

In this section we investigate the translation-translation space in detail, and provide a number of efficient special-purpose algorithms for shape design in this space.

First, we note that the problem of design in translation-translation space, where each object has a single translational degree of freedom along an axis, is equivalent to the design problem in which one object is fixed and the other has two degrees of translational freedom. A simple linear coordinate transformation is sufficient to establish this equivalence. In the following discussion, we will assume that one object is fixed and the other has two degrees of translational freedom, and that the appropriate coordinate transformations have been made. In addition, we assume without loss of generality that the coordinate frame of the configuration space is identical to the space in which objects move (the real space) and that the reference point of the fixed object lies at the origin.

In the translation-translation space, configuration space boundaries are particularly simple: they have the same geometric complexity as the boundaries of the objects. Polygonal objects, in particular, produce configuration spaces whose boundaries are line segments. The geometric routines that handle these spaces (segment intersection, computation of configuration space, etc.) are particularly simple and well understood. We can thus focus on the combinatorial aspect of the algorithms and use standard geometrical techniques described elsewhere to deal with the geometrical aspects ([Preparata85]). In the following, we assume that objects are polygonal.

5.1 Properties of the Configuration Space

Following Lozano-Pérez, we summarize the properties of translation-translation configuration spaces [Lozano-Pérez 83]. Let $\mathcal{R}^2$ be a two-dimensional Euclidean space. a, b, x denote points in $\mathcal{R}^2$, as well as the corresponding vectors. Objects A and B denote (for the purposes of this section) sets of points. Assume that object A is fixed in the plane, and B has 2 degrees of translational freedom. Let $Cspace_B$ be the space of configurations of B, and x a point in it. B in configuration x is denoted by $(B)_x$; B in its initial configuration is denoted by $(B)_0$. The $Cspace_B$ obstacle due to A denoted $OBST(A, B)$ is defined as:

$$OBST(A, B) = \{x \in Cspace_B | A \cap (B)_x \neq \emptyset\} \tag{1}$$

When the coordinate frame of the configuration space is identical to the space in which objects move (the real space), the set of free configurations is the complement of $OBST(A, B)$. Let $A \ominus B = \{a - b \mid a \in A,\ b \in B\}$ be the vector difference (or Minkowski difference of A and B. Then,

Theorem: For A and B sets in $\mathcal{R}^2$, $OBST(A, B) = A \ominus\ (B)_0$. Therefore, the configuration space obstacle consists of the set difference (or vector difference) of A and B in its initial position. When both A and B are convex, the configuration space $CO(A, B)$ can be computed in time linear to the sum of edges of A and B. In general, $CO(A, B)$ can be computed in $O(nm)$, where n and m are the number of edges of A and B respectively.

5.2 Configuration Space Boundaries

There are four types of contacts between two polygons: vertex-vertex, vertex-edge, edge-vertex and edge-edge contact. The vertex-vertex contact can be considered to be a special case of the edge-vertex contact. In addition, note that the edge-vertex and the vertex-edge contacts are entirely symmetrical. There are thus two types of contacts to analyze: vertex-edge contact and edge-edge contact.

Let F be a fixed cartesian coordinate frame. Assume that A is a fixed object, and B is a moving object. Let O_A and O_B be their reference points, and let the contours be described by two sets of points $\{P_i^A\}$ and $\{P_j^B\}$ respectively, corresponding to the vertices of the objects. The coordinates of these vertices are relative to the reference points of each object.

Edge-Vertex Contact

Let $L_A = (P_1^A, P_2^A)$ be an edge of A in contact with a vertex of B, P_1^B. The configuration space boundary created by this contact is the line $L_C = (P_1^C, P_2^C)$ with endpoints (see Figure 5.1):

$$P_1^C = (P_1^A + O_A) - P_1^B \tag{2}$$

$$P_2^C = (P_2^A + O_A) - P_1^B \tag{3}$$

In addition, since we assumed that the coordinate system of the configuration space is identical to the coordinate system of the real space, the edge L_A is parallel to the line L_C.

The analysis of an edge-vertex contact consists of computing the line L_C for two given features, L_A and P_1^B. This line is uniquely determined by the coordinates of the two features. The design of an edge-vertex contact consists of finding, for a given line L_C in the desired configuration space $R(A, B)$, the edge L_A and the vertex P_1^B that create it. This corresponds to finding the coordinates of three points (6 unknowns) in equations (2) and (3) given two points (4 values) and a parallelism constraint. Therefore, in the case of design, the system of equations that determines two features to create an edge-vertex contact is under constrained. An additional value or constraint is necessary to uniquely determine both features.

Edge-Edge Contact

Let $L_A = (P_1^A, P_2^A)$ be an edge of A in contact with an edge of $L_B = (P_1^B, P_2^B)$ of B. The configuration space boundary created by this contact is the line $L_C = (P_1^C, P_2^C)$ is defined as (see Figure 5.1):

$$P_1^C = (P_1^A + O_A) - P_2^B \tag{4}$$

$$P_2^C = (P_2^A + O_A) - P_1^B \tag{5}$$

In addition, the three lines L_C, L_B and L_A must be parallel.

As in the previous case, the analysis of an edge-edge contact consists of computing the line L_C for two given features, L_A and L_B. This line is uniquely determined by the coordinates of the two features. The design of an edge-edge contact consists of finding, for a given line L_C in the desired configuration space $R(A, B)$, the two features L_A and L_B that create it. This corresponds to finding the coordinates of four points (8 unknowns) in equations (3) and (4) given two points (4 values) and a parallelism constraint. Therefore, in the case of design, the system of equations that determines two features to create an edge-edge contact is also under constrained. Three additional values or constraints uniquely determine both features.

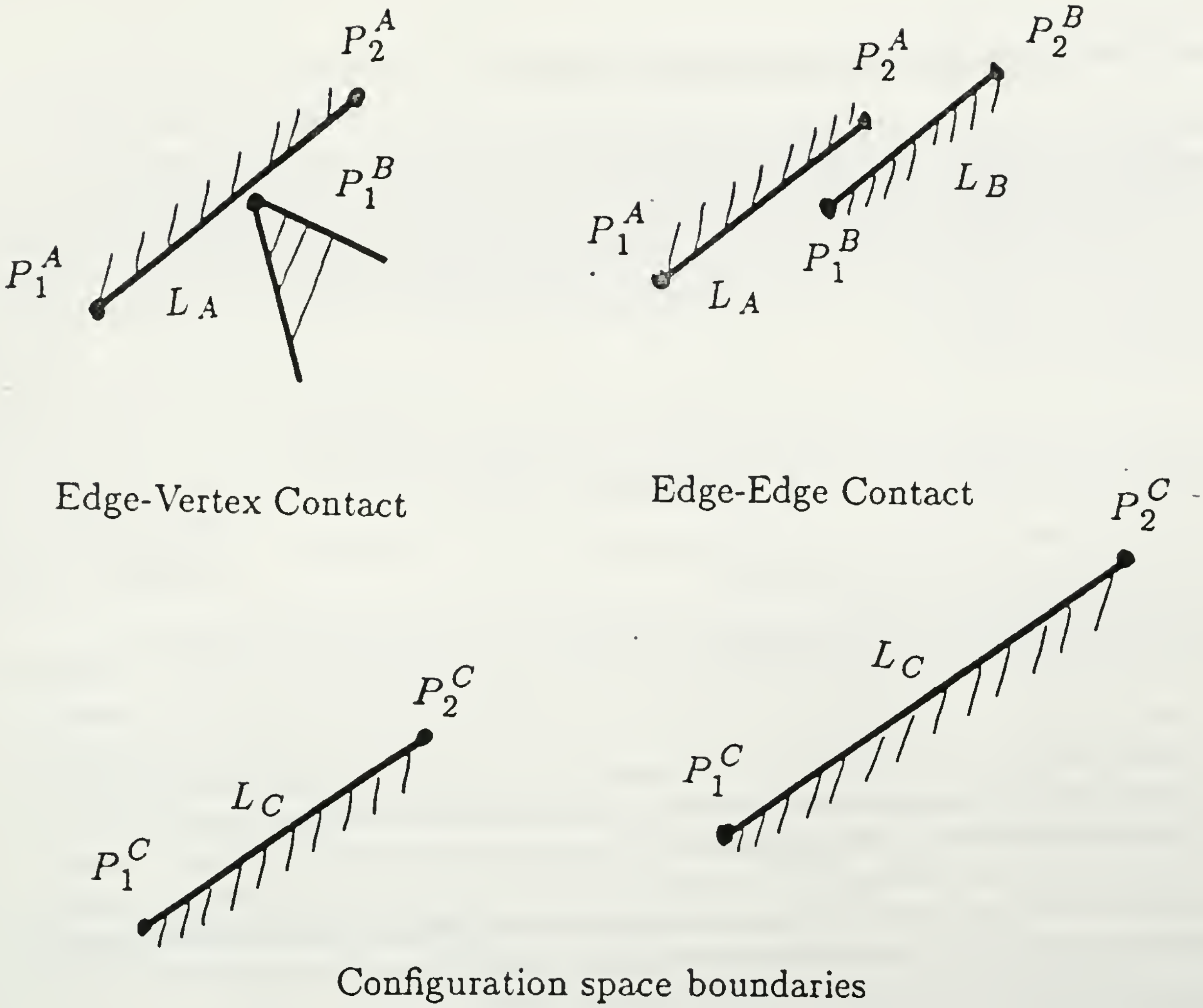


Figure 5.1: Edge-Vertex Contact and Edge-Edge Contact in the Translation-Translation Space.

5.3 Designing with the general algorithm

Consider again the general backtracking algorithm presented in section 3. For a given configuration space boundary of $R(A, B)$ that is not in $CO(A, B)$, we have to produce one or two object features that create this boundary. By the analysis of the previous subsection, there are two contacts that create a configuration space boundary: an edge-vertex contact and an edge-edge contact. In both cases, the system of equations that defines the two features is under constrained. Thus, an additional constraint must be introduced. Here is a list of alternatives:

Edge-Vertex Contact

1. Pick an existing vertex in B and compute the two endpoints of the edge L_A.
2. Pick an existing edge in A and find a vertex P_1^B.
3. Pick an existing vertex in A, say P_1^A, and compute the vertices P_2^A and P_1^B. This produces a edge that is connected to the boundary of A.
4. Pick a direction or a length for the edge L_A. This introduces an extra constraint which thus determines P_1^A, P_2^A and P_1^B.
5. Find additional constraints that restrict the choices on P_1^A, P_2^A, P_1^B. The maximum and minimum length of edge L_A, maximum and minimum distance of the new vertex P_1^B from the existing boundary of B are some examples of such constraints. All constraints should be based on information about the desired boundary L_C and its immediate configuration space boundary neighbors.

Note that for the first three modifications, a different existing feature is chosen. so the number of choices is bounded by the number of features of A and B. Modifications that require the introduction of all three vertices P_1^A, P_2^A, P_1^B lead to an infinite number of choices.

Edge-Edge Contact

1. Pick two existing vertices, one of A and one of B.
2. Pick an existing edge of either A or B.
3. Pick an edge of A and a vertex of B (or the opposite).
4. Pick two vertices, one from L_A and one from L_B.
5. Pick two edge lengths for L_A and L_B, a vertex of one of the edges. This introduces three more constraints and thus uniquely determines P_1^A, P_2^A, P_1^B and P_2^B.
6. Find additional constraints that restrict the choices on $P_1^A, P_2^A, P_1^B, P_2^B$. The maximum and minimum length of edges L_A and L_B, maximum and minimum distance of the endpoints of L_A and L_B from their corresponding existing contours are some examples of such constraints. All constraints should be based on information about the desired boundary L_C and its immediate configuration space boundary neighbors.

As in the previous case, the first four modifications have a finite number of choices, and the last two have an infinite number of choices.

If one of the objects has a fixed shape, i.e is its contour is not allowed to change, then the last two cases in both types of contact do not apply. Therefore, the number of choices is finite, and the complexity of the backtracking algorithm is exponential in the number of choices. By introducing restrictions on the shape of the objects, a number of more efficient algorithms can be developed. The next sections present restricted design problems and the algorithms for their solution.

5.4 Two Convex Objects

Here, we assume that both objects are simple convex polygons, and thus their configuration space is also a convex polygon.

Problem 1:

Given a simple convex polygon A and a convex configuration space $R(A,B)$, find a simple convex polygon B such that $CO(A,B) = R(A,B)$ without altering A. Assume that the contours of A and the boundary of $R(A,B)$ are given in clockwise order, and that B is a fixed object.

Algorithm 5.1:

The algorithm proceeds by "walking" clockwise around the contour of both A and $R(A,B)$ simultaneously. Since both objects are convex, there is a single candidate feature for A to produce the next configuration space boundary (Convexity Theorem, section 4). The edges of B are added clockwise. At the end of this process, the consistency of B is tested by verifying that the first edge added meets the last edge at a vertex (i.e. the shape of B is closed). The solution of this design problem, if it exists, is unique.

Let "create-edge(f_A, f_R)" be a procedure whose input is a feature (a vertex or an edge) of A and a line of $R(A,B)$ and whose output is the corresponding edge of B determined by the equations defined in the previous section. Similarly, "create-point(P_A, P_R)" returns the vertex of B corresponding to the vertex P_A of A and the point P_R of $R(A,B)$. Let "next(f)" be a procedure that returns the feature adjacent to f in clockwise order, and let current(A), current(B) and current(R) be three pointers to a feature in A, B and $R(A,B)$ respectively. See the procedure DESIGN-CONVEX.

Since the local check in step 2.4 takes constant time, the complexity of the algorithm is linear in the number of features of $R(A,B)$. Note that if we are given an initial shape for B, we can simply disregard it and run the above procedure to produce a new B.

Problem 2

Given a convex configuration space $R(A,B)$, find two simple convex objects A and B such that $CO(A,B) = R(A,B)$.

This problem is under constrained, since and thus admits infinitely many solutions. One possible solution is to set A to be half the size of $R(A,B)$ (i.e reduce the length of all the lines in $R(A,B)$ by half) and B to be half the size of $-R(A,B)$ (vector difference). This guarantees that both A and B are simple convex objects, and that $CO(A,B) = R(A,B)$. The solution, however, might violate some of the design

Procedure DESIGN-CONVEX($A, R(A,B)$)

1. Sweep a line and find the leftmost point in A and $R(A,B)$. Both points uniquely determine a starting point in B.
   current(A) := leftmost(A)
   current(R) := leftmost($R(A,B)$)
   current(B) := create-point(current(A), current(B))

2. For all boundaries of $R(A,B)$, in clockwise order, do

   2.1. current(R) := next(current(R))

   2.2. If current(R) is not parallel to next(current(A)) then

   If current(R) = create-line(next(current(A)), current(B))
   then current(A) := next(current(A))
   else current(B) := create-edge(current(A), current(R))

   2.3. Else (current(R) is not parallel to next(current(A)))

   If length(current(A)) $<$ length(current(R))
   then current(B) := a edge parallel to current(R), whose length is the difference of the two lengths.
   current(A) := next(current(A))

   2.4. If current(B) changed, then check local design constraints. If any of them is violated, return 'fail.

3. End

4. If last(B) is not adjacent to first(B) then return 'fail

5. Else return(B)

constrains. The complexity of this method is again linear in the size of $R(A,B)$.

5.5 One convex object, one concave object

Here, we assume that one object, A, is a simple concave polygon and B is a simple convex polygon. The configuration spaces $CO(A,B)$ and $R(A,B)$ are possibly concave, depending on the objects.

Problem 3

Given a concave object A and a configuration space $R(A,B)$, find a convex polygon B such that $CO(A,B) = R(A,B)$ without altering A.

Algorithm 5.3

The solution is based on the use of convex hulls ([Preparata85]). Object B is computed by taking the convex hull of both $R(A,B)$ and A and then using Algorithm 5.1. The result of algorithm 5.1 is the convex hull of B, which is equal to B since we assumed that B is convex. Following is the proof that this method is correct.

Proof

The boundary of $R(A, B)$ constitutes also the boundary of the configuration space obstacle (as defined in 5.1), $OBST(A, B)$. We have that $OBST(A, B) = A \ominus (B)_0$. Since we will always refer to B in its initial position, we will simply write $OBST(A, B) = A \ominus B$. Since B is convex, $convex\_hull(B) = B$. By the properties of the convex hull operation, $convex\_hull(OBST(A, B)) = convex\_hull(A \ominus B)$.
In order to prove that the above algorithm is correct, we need to prove that, when B is a convex object,

$$convex\_hull(A \ominus B) = convex\_hull(A) \ominus B \tag{6}$$

If this equality holds, then the boundary of B is uniquely determined. To see this, consider the boundaries of A and B defined as a set of vectors from the origin to their corresponding vertices. Let "-" be the vector difference operation. Then, equation (6) can be rewritten as: $B = convex\_hull(A) - convex\_hull(OBST(A, B))$. Therefore, to find B, we take the convex hull of A, the convex hull of $CO(A, B)$ and then use Algorithm 5.1. We now prove (6) in two steps:

1) $convex\_hull(A) \ominus B \subseteq convex\_hull(A \ominus B)$:
Let a_1, a_2 be two points in A, and let b be a point in B. By the definition of convex hull, for all α in (0, 1), the point $p = [\alpha a_1 + (1 - \alpha)a_2] - b$ is in $convex\_hull(A) \ominus B$. By rewriting b as $\alpha b + (1 - \alpha)b$ and substituting it into p we get $p = \alpha(a_1 - b) + (1 - \alpha)(a_2 - b)$. Now both points $p_1 = (a_1 - b)$ and $p_2 = (a_2 - b)$ belong to $A \ominus B$. Therefore, the point $p = \alpha p_1 + (1 - \alpha)p_2$ also belongs to $convex\_hull(A \ominus B)$.

2) $convex\_hull(A \ominus B) \subseteq convex\_hull(A) \ominus B$:
Let p_1, p_2 be two points in $A \ominus B$ such that $p_1 = a_1 - b_1$ and $p_2 = a_2 - b_2$, where $a_1, a_2 \in A$ and $b_1, b_2 \in B$. Then, by the definition of convex hull,

$$p = \alpha p_1 + (1 - \alpha)p_2 = \alpha(a_1 - b_1) + (1 - \alpha)(a_2 - b_2) \tag{7}$$

is in $convex\_hull(A \ominus B)$. By rewriting (7) we get

$$p = [\alpha a_1 + (1 - \alpha)a_2] - [\alpha b_1 + (1 - \alpha)b_2)]$$

The first portion belongs to $convex\_hull(A)$, and the second to $convex\_hull(B) = B$. Therefore p also belongs to $convex\_hull(A) \ominus B$.

Taking the convex hull of k points in the plane can be done in time $O(n \log n)$. Therefore, the complexity of algorithm 5.3 is $O(k \log k)$, where $k = n + m$ is the number of features in $R(A, B)$.

Problem 4

Given a convex object A and a possibly concave configuration space $R(A, B)$, find a concave object B such that $CO(A, B) = R(A, B)$.

Algorithm 5.4:

This algorithm is very similar to Algorithm 5.1, except that we can no longer take the next feature in the contour of A as we design B (step 2.2). Instead, we find the next feature of A by sweeping a line parallel to the current segment L_C of $R(A, B)$. The first vertex v_A of A to come in contact with the sweep line is the vertex that, together with an edge of B, produces L_C. Given v_A, the coordinates of the two endpoints of the new edge of B can be directly determined by equations (2) and (3). This method produces a disconnected boundary of B that is then connected to B by using parts of the contours of A to fill in the missing contour.

Let "sweep-line(L_R)" be a function that, when given a line L_R of $R(A,B)$, and a direction of sweep returns the first feature of A to come in contact with the sweep of L_R (see Figure 5.2 for an example). We substitute Step 2.2 in Algorithm 5.1 by the following step:

2.2. If current(R) is not parallel to sweep-line(current(R)) then

If current(R) = create-edge(sweep-line(current(R)), current(B))
then current(A) := sweep-line(current(R))
else current(B) := create-edge(sweep-line(current(R)), current(R))

In addition, a new step (between step 3 and 4) is introduced to complete the boundary of B, using the contour of A. See Figure 5.2 for an example on how object B is completed by using part of the contour of S. Note that Algorithm 5.1 is a special case of this algorithm, since the function sweep-line(current(R)) is always equal to next(A).

5.6 The General Method

For the general case of design in the translation-translation space, we allow both objects to be concave, and not necessarily simple (i.e they can contain holes). The configuration spaces $CO(A,B)$ and $R(A,B)$ can thus consist of disconnected regions. In the following, we will assume that the shape of one object, A, is fixed. The goal is to find the largest (in a set-theoretical sense) object B such that $CO(A,B) = R(A,B)$.

Note that we can no longer use Algorithm 5.2, since the sweeping of a line of $R(A,B)$ no longer identifies the feature of A that created it. Instead, we use a method that "carves" B by positioning A in the free placements defined by $R(A,B)$. We start by taking B to be the complement of A. In this case, the configuration space $CO(A,B)$ is reduced to a single point, corresponding to the only free placement of B. By sweeping A along several lines corresponding to free placements inside $R(A,B)$, we "cut" B to allow the sweep. This is done by translating A in the direction and distance of the line and then intersecting the trace with B from the initial to the final positions. The resulting object (possibly disconnected) is guaranteed, by construction, to be able to translate along this line. In order to cover the whole space, we triangulate $R(A,B)$ and then sweep along the edges of each triangle. Note that when sweeping an object A along the edges of a triangle, two disconnected regions might be created (see Figure 5.3). One of this regions must be discarded since it contains forbidden placements that are in reality free. To determine which region must be discarded, it is sufficient to pick one free placement inside the triangle of $R(A,B)$ and determine to which region it belongs. The other region is then discarded. If the design is possible, then the resulting B matches the requirements. Otherwise, the design problem has no solution.

Since this algorithm produces the largest set such that $CO(A,B) = R(A,B)$, the designed object B may in fact contain several pieces, some of which are redundant. Also, the designed object is not necessarily the simplest object.

Complexity

Let the complexity (number of edges) of A be n, of B be m and of $R(A,B)$ be k. The triangulation of $R(A,B)$ in step 1 produces $O(k)$ triangles. For each triangle, the computation of the sweep of A can produce at most $O(n^2)$ pieces for B. To see this, consider the object A in Figure 5.4. It consists of two "combs" having n long, thin "teeth" so that their "backbones" are perpendicular to one another. The translation of

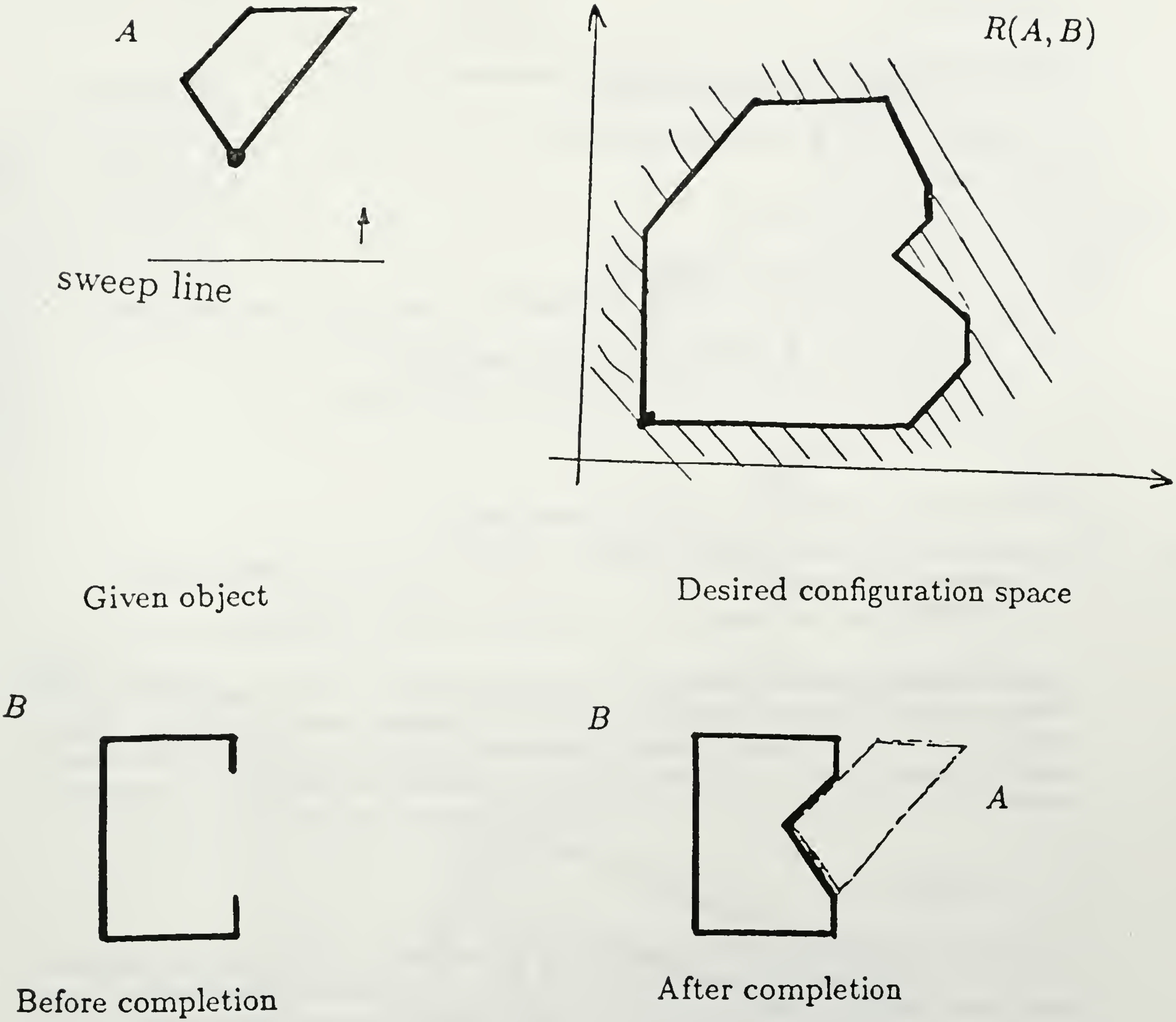


Figure 5.2: Example of Object Completion in Algorithm 5.3.

Algorithm 5.5:

Procedure DESIGN-BY-SWEEP($A, R(A, B)$)

1. Triangulate $R(A, B)$ by introducing new edges.
2. B := complement(A).
3. For every triangle in $R(A, B)$ do
   - 3.1 For every edge c_i in the triangle, do:
     - a. Translate A in the direction and length of c_i, recording the traces made by the vertices of A.
     - b. Intersect this trace with B and produce a contour for B.
   - 3.2 If this sweep procedure creates two regions, discard one of them.
   - 3.3 End.
4. End
5. Merge the pieces of B created by the sweep in each triangle.
6. Compute the configuration space, $CO(A, B)$. If it is equal to $R(A, B)$ and no design constraints are violated, then return(B).
   Else return 'fail.

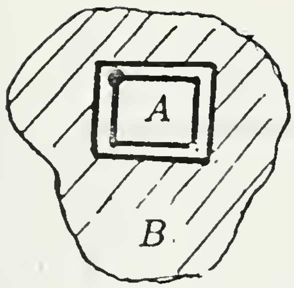
---

this object along the axis x and y produces at most a quadratic number of isolated pieces. Since there are k triangles, step 3 of the algorithm produces at most $O(n^2k)$ polygonal pieces for B. Merging a polygonal regions in the plane can be done in $O(a \log a)$ ([Preparata85]). Since we have at most $O(n^2k)$ regions to consider for mergin in step 5, $a = n^2k$, the complexity of this step is $O(n^2k \log n^2k)$. This dominates the complexity of all the other steps. Thus the overall complexity of the algorithm is $O(n^2k \log n^2k)$.

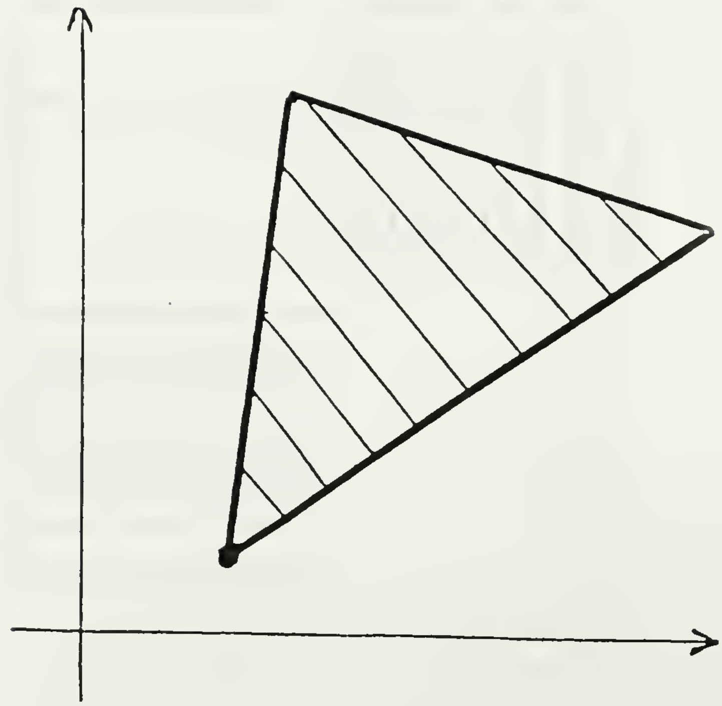
5.7 Evaluation

The complexity of the special purpose algorithms for the translation-translation space are significantly better than the general algorithm presented in section 3. Note that when the ordering heuristics in the general backtracking algorithm are used, and both objects are simple convex polygons, the algorithm runs in time linearly proportional to the size of $R(A, B)$, just as Algorithm 5.1 in this section. This is because the ordering heuristics guarantee that first candidate feature chosen is always the right one, precluding the need for backtracking (see also section 3.4).

The algorithms presented here can easily be extended to include arcs in the object boundary. The combinatorial component of the algorithms remains unchanged, and the geometric component is extended to deal with arcs.



Initial Shapes



Desired configuration space

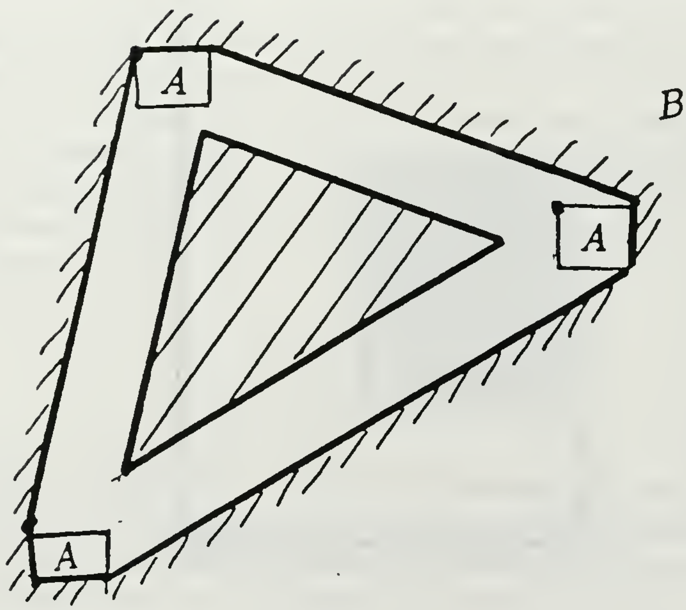


Figure 5.3: The Two Regions Created by Sweeping an Object A. The Center Region Must be Eliminated Since it Includes Free Placements of A.

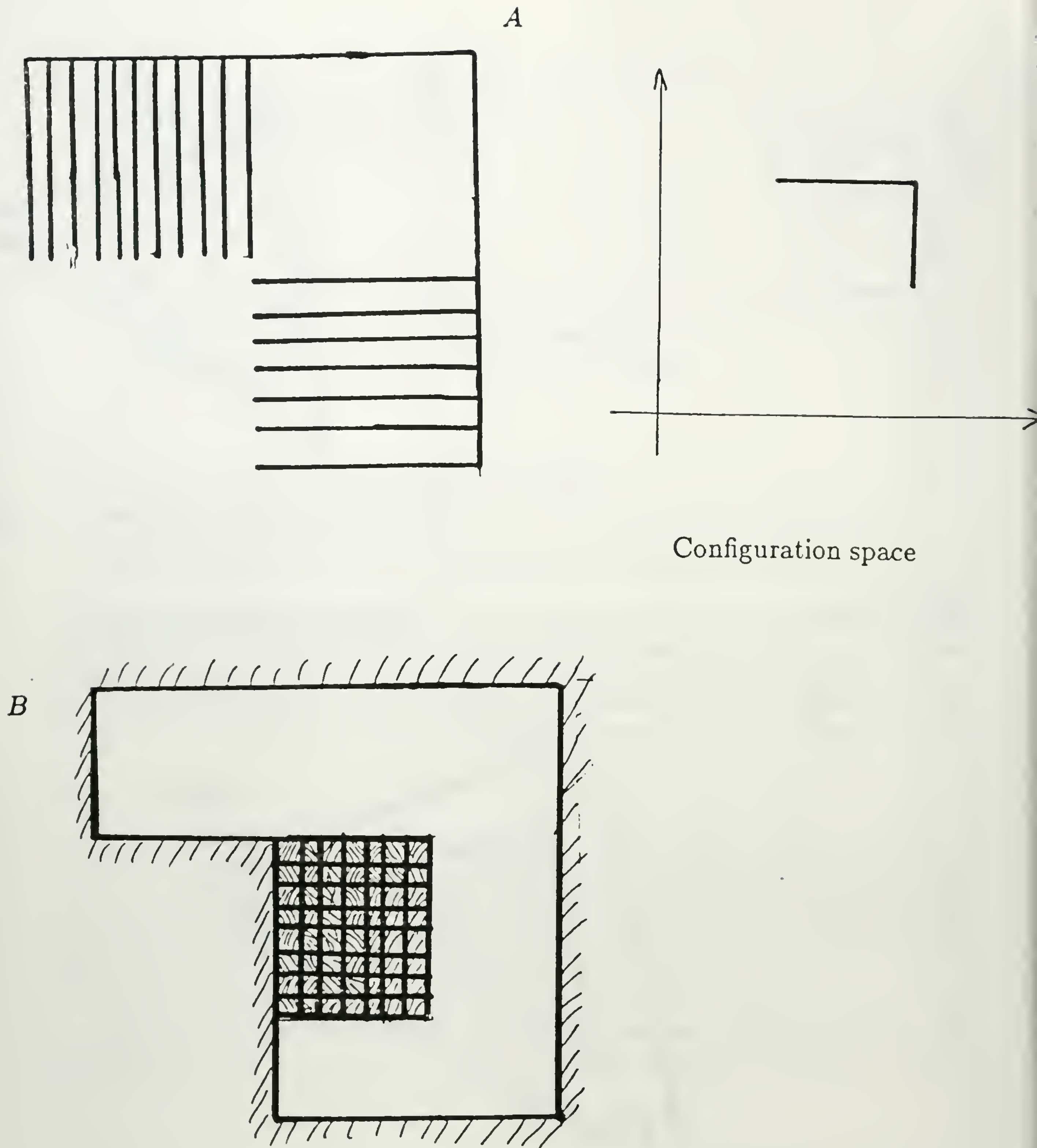


Figure 5.4: An Object A and a Configuration Space that Produce a Quadratic Number of Disconected Pieces for B.

6 Qualitative Shape Design

Up to now, we assumed that we either have, or can produce, an exact description of the desired configuration space. In some cases, such a precise description is not available, or not required.

Consider the following example: we are given a disk A that can rotate around axis O_1 and a rectangle B that can translate along axis O_2. Let θ and X be their rotation and translation parameters, respectively. Suppose we want, for a full rotation of A, B to slide up, then down, and then stay stationary. The precise relationship between X and θ is not important. We only require X to increase when θ increases for the intervals $X \in [0, X_0]$ and $\theta \in [0, \pi/2]$, and X to decrease when θ increases for $X \in [X_0, 0]$ and $\theta \in [\pi/2, \pi]$. For $\theta \in (\pi, 2\pi)$, X is to remain constant, $X = 0$. This description is not sufficient to produce an exact configuration space since the type of configuration space boundary in the first two regions is unknown. Indeed, *any* boundary is satisfactory as long as the qualitative relations between the parameters hold continuously in each region. Figure 6.1 shows a solution that meets these requirements. The given boundary points are matched exactly, but also new boundary points are introduced.

To design shapes from qualitative descriptions, we no longer require an exact boundary match between $CO(A, B)$ and $R(A, B)$. The matching requirement for qualitative boundaries is relaxed as follows: let S be a set of boundary segments of $CO(A, B)$. S matches a qualitative boundary defined by two given points P_1 and P_2 of $R(A, B)$ iff:

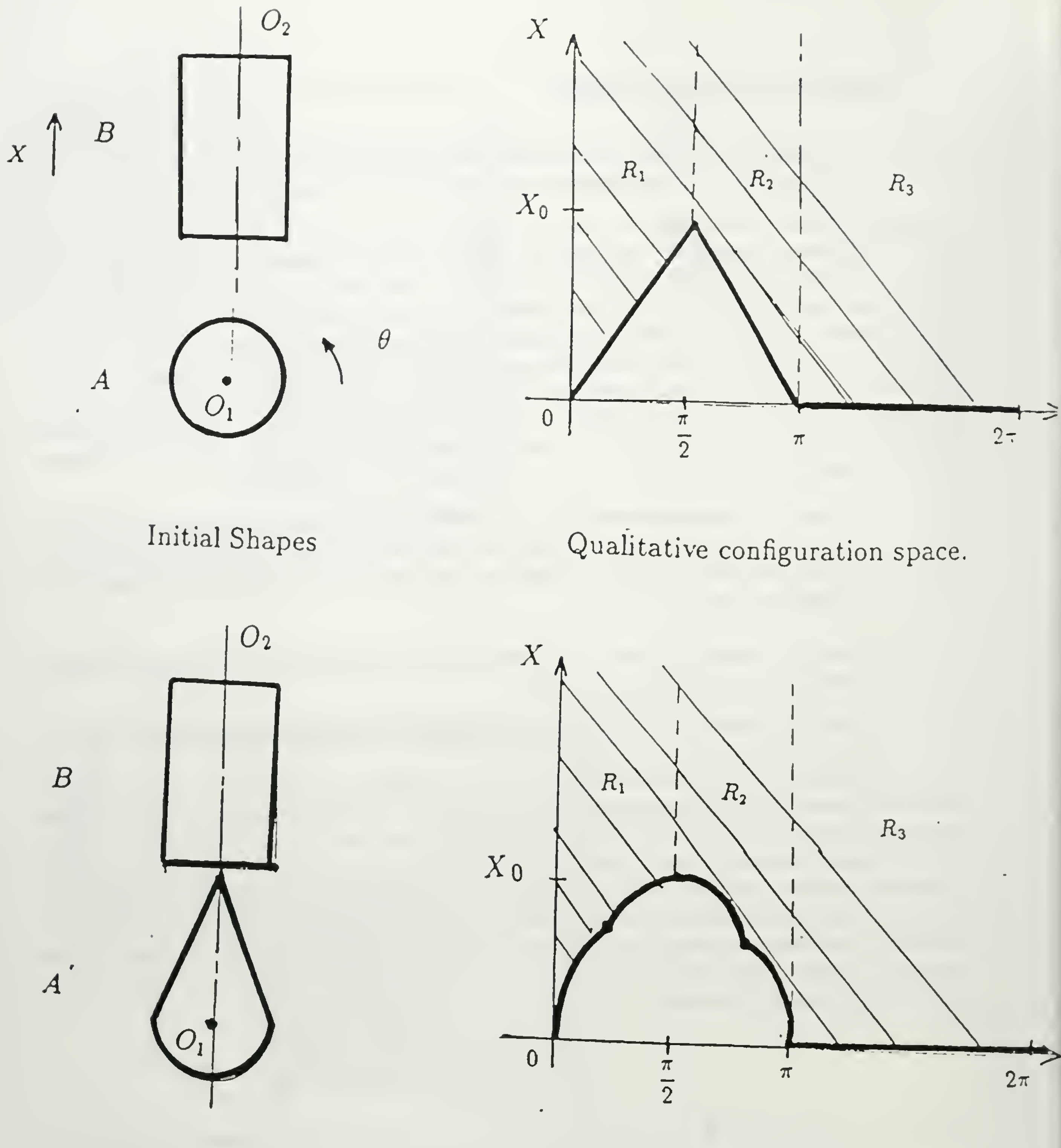
1. The boundary segments of S form a connected, piecewise differentiable boundary whose endpoints are P_1 and P_2.
2. Each boundary segment in S reflects the same qualitative change than the change from P_1 to P_2.

Defining a qualitative motion relation between two objects allows us to broaden the number of choices of pairwise contacts. The chief guideline in the selection process is to obtain a configuration space boundary that is monotonically increasing or decreasing, according to the relative positions of the endpoints defined by the parameter intervals. In order to determine how to choose a pair of features that will produce such a boundary, we need to store additional information in the elementary contact table that will indicate for what value range the configuration space boundary will be monotonically increasing or decreasing. For example, in the translation-translation space, the line boundary is increasing when the angle α of the boundary line with respect to the coordinate system is between 0 and $\pi/2$, that is, the tangent of the configuration space line is positive:

$$\tan\alpha = \frac{y_1^C - y_2^C}{x_1^C - x_2^C} \geq 0$$

where $Q_1 = (x_1^C, y_1^C)$ and $Q_2 = (x_2^C, y_2^C)$ are new points of the configuration space boundaries (not necessarily equal to P_1 and P_2). This relation imposes a set of constraints on the position of the features that produce this boundary. In other spaces, there might be additional constraints on the possible positions of Q_1 and Q_2. All these constraints are included in the table of elementary interactions.

A qualitative configuration space boundary is defined by two endpoints. The first attempt to match it will be to look for a pair of features that creates a boundary connecting the two given endpoints. If this is



Initial Shapes

Qualitative configuration space.

Modified Shapes

An acceptable configuration space

Figure 6.1: A Qualitative Boundary Description, and an Acceptable Solution for it.

not possible, we need to introduce new points to break the boundary into several pieces. Borrowing Kuipers' terminology [Kuipers 84] we call these new points *landmark points*. In order to keep the design as simple as possible, new landmark points should be introduced only when it is impossible to design without them. The following backtracking recursive algorithm introduces new landmark points only when necessary: Let *Boundary-Sign*(B) be a function that returns "+" if the boundary B is continuously increasing, "0" if it is constant and "−" if it is decreasing. Let Q_{change} be sign of change of the qualitative boundary.

Procedure FIND-Q-BOUNDARY (P_1, P_2, Q_{change})

1. Find a pair of features that produce a boundary B from P_1 to P_2, such that *Boundary-Sign*$(B) = Q_{change}$.
2. If such a boundary is impossible, choose a pair of features that produce a continuous boundary B from P_1 to an intermediate feasible point Q. This new point should be as close as possible to P_2.
   When choosing features, prefer the features that form a continuous boundary in both objects.
   In addition, *Boundary-Sign*$(B) = Q_{change}$.
3. If no point Q can be found, return fail.
   Else call *FIND-Q-BOUNDARY* (Q, P_2, Q_{change}) recursively.

---

7 Shape Design from Causal Descriptions

As mentioned in section 2, an alternative way of describing the behavior of a mechanism is a causal description. A causal description characterizes the effect that a sequence of input motions applied to one object in the mechanism have on the rest of the objects. As an example consider the two gears shown in Figure 7.1 (the teeth of the gears are schematically represented by segments on the circumference of a circle). The first gear, A, is fully toothed, while the second, B, has teeth only in half of its circumference. The radius of both gears is equal. An example of a causal description for this kinematic pairs is as follows: the clockwise rotation of B causes a counter clockwise, intermittent rotation of A. A continuous clockwise rotation of A causes a counter clockwise rotation of B by π and then no further motion of B. The speeds of rotation are equal.

Causal descriptions of kinematic behavior are sometimes preferable and more intuitive than descriptions in terms of possible motions or configuration spaces. Causal descriptions correspond to the notion of the *state diagram* of a device, originally proposed in [DeKleer76], and extensively used in Qualitative Physics [DeKleer84], [Forbus84], [Kuipers84]. State diagrams for mechanical devices are used in [Faltings87] and [Joskowicz87a]. A state diagram is a directed graph in which every node represents a qualitatively different behavior of the mechanism resulting from a sequence of input motions. Edges represent transitions between behaviors.

As shown in [Joskowicz87a], there is a direct relationship between the region diagram of a mechanism (as defined in section 2.2) and the state diagram for given sequence of input motions. Given a region diagram and a sequence of input motions, the corresponding state diagram is derived using a motion propagation procedure. The difference between state diagrams and region diagrams is that the state diagram describes only one of many behaviors, whereas the region diagram describes all the possible behaviors. Therefore, a set of state diagrams might be necessary to account for all the qualitatively different behaviors of a mechanism. For example, in order to have a complete causal description of the gear pair of Figure 7.1, we have to complete the previous causal description by stating what happens when B rotates counterclockwise and when A rotates counterclockwise. Also, we should state that no other qualitatively different behavior is possible. Indeed, a causal description can be interpreted as either being a *partial* or a *complete* description of the desired behavior. Both descriptions require the described behaviors to take place, but the partial description allows additional qualitatively different behaviors. A complete description requires that no other qualitatively different behaviors take place. In both cases, the design is considered successful when the input motion sequences applied to the objects produce exactly the original state diagrams.

Let $S = \{S_1, \ldots, S_n\}$ be a collection of state diagrams, where each state diagram S_i is a triple $[\sigma_i, \{s_{ij}\}, \{< s_{ij}, s_{ik} >\}]$. σ is the input motion sequence, $\{s_{ij}\}$ is the set of states describing the motion of each object, and $\{< s_{ij}, s_{ik} >\}$ is the set of state transitions. The function $apply(\sigma, CO(A, B))$ produces the state diagram corresponding to the input sequence σ and the configuration space $CO(A, B)$ (the procedure to compute *apply* is described in [Joskowicz87a]). The shapes of A and B *satisfy* a given collection S of state diagrams iff:

$$\forall S_i \in S \ \wedge \ \forall \sigma_i \in S_i, \ \ apply(\sigma_i, \ CO(A, B)) = S_i$$

i.e., the application of each input motion sequence to the actual configuration space produces the same state diagram as the one desired one. A configuration space that satisfies the above property is *acceptable*. For a given set of state diagrams, the goal is to construct an acceptable desired configuration space, $R(A, B)$.

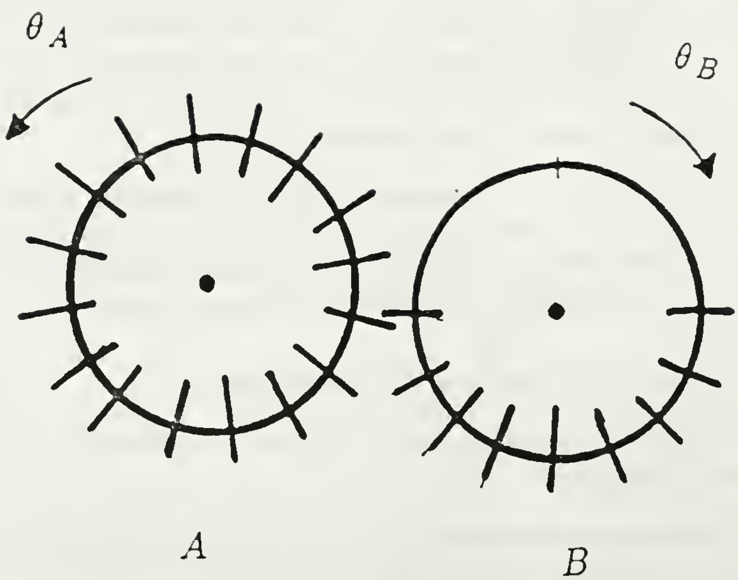


Figure 7.1: The Half Gear Pair Example.

7.1 Deducing the Region Diagram from State Diagrams

Given a set of state diagrams $\{S_1, ..., S_n\}$, our goal is to construct a region diagram (and thus a configuration space) $R(A, B)$ that satisfies the acceptability criterion. Since there may be many region diagrams for which the acceptability criterion is satisfied (especially when the state diagrams describe only a subset of the possible behaviors), we want to produce the "weakest" or *least constrained* region diagram that satisfies the acceptability criterion. For two given region diagrams $R(A, B)$ and $R'(A, B)$ that satisfy the acceptability criteria for S, $R(A, B)$ is said to be less constrained than $R'(A, B)$ iff the set of free placements in $R(A, B)$ includes the set of free placements in $R'(A, B)$.

We construct $R(A, B)$ by composing individual configuration spaces $R_i(A, B)$ resulting from each S_i. The space $R_i(A, B)$ is in turn constructed by composing configuration space regions r_{ij} resulting from each state s_{ij}. Each state s_{ij} is mapped into a region of the configuration space by using the information contained in the state about object motions and their relationships:

1. The type of motions determines the design space.
2. The intervals of the motion parameters determine the region of the configuration space in which the behavior takes place.
3. The boundary of the configuration space is determined either by an explicitly given relation ($X_A \geq f(X_B)$), or deduced from the causal description that defines the instigator of the movement and the direction of change for the motion parameters:
   motion(A) CAUSES *motion*(B), *direction*(X_A), *direction*(X_B),

The configuration space boundary resulting from a causal description is a qualitative boundary, whose endpoints are determined by the intervals of X_A and X_B. The region of free placements is determined by one of the eight possible combinations of values for *direction*(X_A), *direction*(X_B) and *motion*(A) CAUSES *motion*(B), as shown in Figure 7.2. For example, in the first case, the qualitative configuration space boundary is defined by the endpoints (X_1^A, X_1^B) and (X_2^A, X_2^B). The set of free placements corresponds to the region $X_B \leq f(X_A)$, where f is the equation of the boundary line.

For every state s_{ij}, we construct the corresponding region r_{ij}. The regions r_{ij}, $1 \leq j \leq n_i$ are then combined to produce $R_i(A, B)$. Regions are combined by taking the union of their forbidden placements. Figure 7.3 shows the merging of two regions, R_1 and R_2: R_1 is defined in the intervals $X_A \in [a_1, b_1]$, $X_B \in [c_1, d_1]$ and the relation $X_A \leq f_1(X_B)$. R_2 is defined in the intervals $X_A \in [a_2, b_2]$, $X_B \in [c_2, d_2]$ and the relation $X_A \geq f_2(X_B)$. Their combination is indicated by the dashed region corresponding to free placements. Conceptually, composing two regions amounts to requiring two behaviors to take place in the interval common to the two regions, and preserving the behaviors in the disjoint subregions. The configuration space resulting from this composition is then partitioned into regions to obtain the region diagram for the described behavior, $R_i(A, B)$. Note that this composition method produces the least constrained region diagram.

Once the region diagram $R(A, B)$ has been constructed, we can follow the design technique described previously to design A and B by matching $R(A, B)$ and $CO(A, B)$. If we require a strict interpretation of the state diagrams, i.e we assume that all the possibly qualitative different behaviors of the pair are

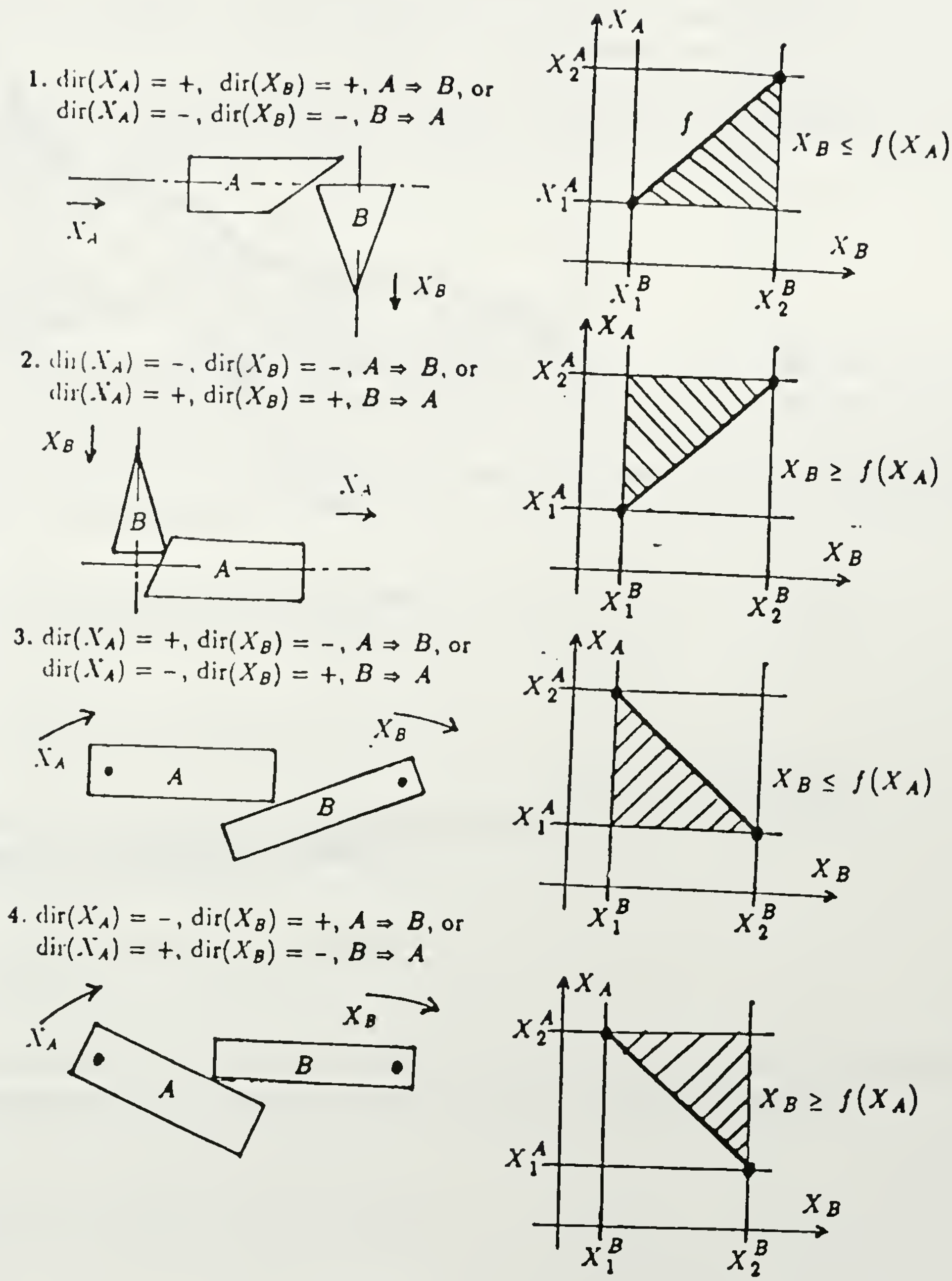


Figure 7.2: Causal Descriptions and their Corresponding Qualitative Configuration Space Regions.

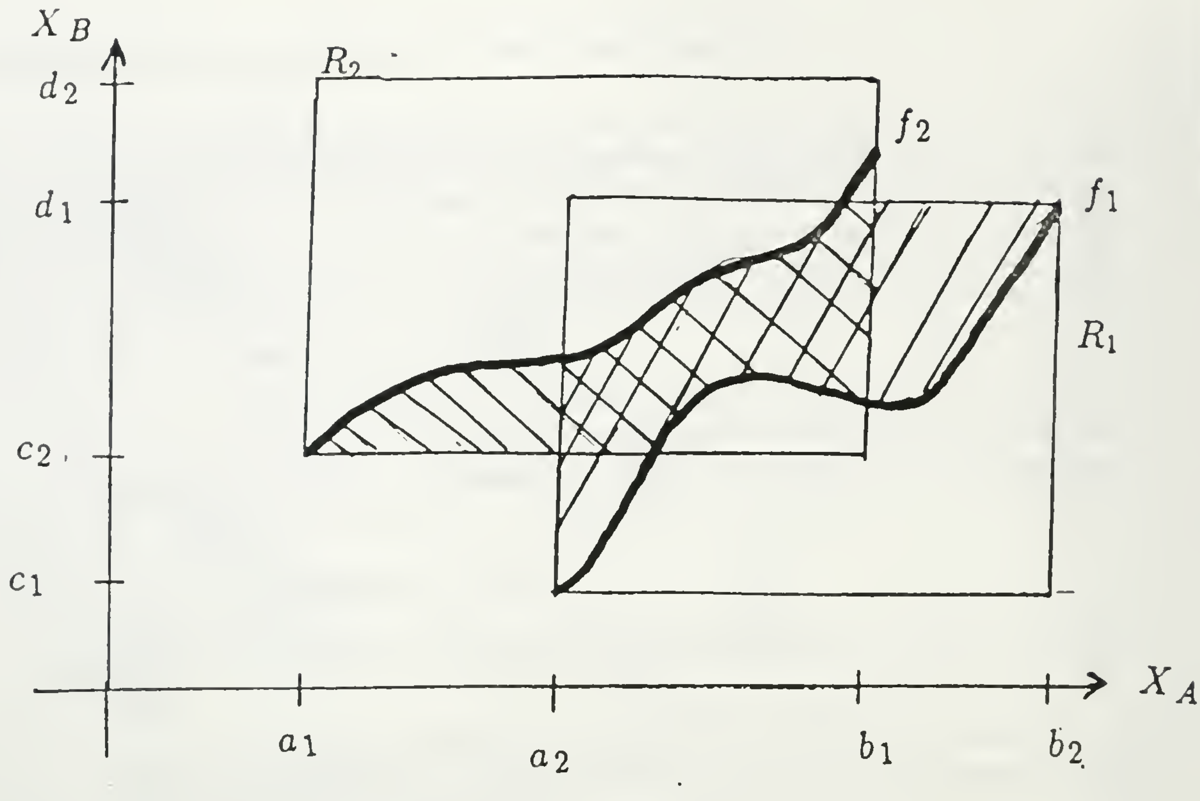


Figure 7.3: Merging Two Region Diagrams, R_1 and R_2.

described by the state diagrams, we will require an exact match between $R(A,B)$ and $CO(A,B)$ (or a qualitative match if the boundaries are qualitative). If the set of state diagrams constitutes a partial description of the pair's behavior, meaning that the pair A,B should have at least the behaviors described by S. but possibly more, then we no longer require an exact match between $R(A,B)$ and $CO(A,B)$. In this case. we permit regions in $CO(A,B)$ that cannot be reached in the description of $R(A,B)$ (disconnected components). Therefore, $R(A,B)$ matches $CO(A,B)$ iff there exist a set of regions r_1, ..., $r_n \subset CO(A,B)$ such that $R(A,B) = r_1 \cup ... \cup r_n$.

7.2 An Example

Consider again the half gear example introduced in Figure 7.1. Suppose we want to design this pair from the description of two state diagrams, S_1 and S_2, as shown in Figure 7.4. In S_1, the input motion is a continuous counterclockwise rotation of B. We want this rotation to cause an intermittent clockwise rotation of A, where both gears turn together at the same speed for an interval of π and then (assuming no inertia) B continues to rotate by another π while A stays fixed. This process is to repeat itself indefinitely.

s_{11}: $rotation(B, O_B, \theta_B)$, $\theta_B \in [0,\pi]$, $direction(\theta_B) = +$
$rotation(A, O_A, \theta_A)$, $\theta_A \in [0,-\pi]$, $direction(\theta_A) = -$
$\theta_A = -\theta_B$

s_{12}: $rotation(B, O_B, \theta_B)$, $\theta_B \in [\pi, 2\pi]$, $direction(\theta_B) = +$
$fixed(A), \theta_A = 0$

s_{13}: $rotation(B, O_B, \theta_B)$, $\theta_B \in [2\pi, 3\pi]$, $direction(\theta_B) = +$
$rotation(A, O_A, \theta_A)$, $\theta_A \in [0,\pi]$, $direction(\theta_A) = -$
$\theta_A = -\theta_B$

s_{14}: $rotation(B, O_B, \theta_B)$, $\theta_B \in [3\pi, 4\pi]$, $direction(\theta_B) = +$
$fixed(A), \theta_A = 0$

In S_2, the input motion is a continuous clockwise rotation of A. We want this rotation to cause a counterclockwise rotation of B by π (at the same speed) and then no further motion of B.

s_{21}: $rotation(A, O_A, \theta_A)$, $\theta_A \in [0,\pi]$, $direction(\theta_A) = +$
$rotation(B, O_B, \theta_B)$, $\theta_B \in [0,-\pi]$, $direction(\theta_B) = -$
$\theta_A = -\theta_B$

s_{22}: $rotation(A, O_A, \theta_A)$, $\theta_A \in [0, 2\pi]$, $direction(\theta_A) = +$
$fixed(B), \theta_B = 0$

The state diagram S_1 consists of three regions, as indicated in Figure 7.5: r_1, r_2 and r_3. The region diagram corresponding to S_2, $R_2(A,B)$ has two regions, r_4 and r_5. The composition of $R_1(A,B)$ and $R_2(A,B)$ is as follows: in the interval $\theta_A \in [\pi, 2\pi]$ and $\theta_B \in [0, 2\pi]$, two regions, r_3 and r_5 are identical and thus belong to $R(A,B)$. In the interval $\theta_A \in [0,\pi]$ and $\theta_B \in [0,\pi]$ there is only one region, $r_2 \subset R(A,B)$. In the interval $\theta_A \in [0,\pi]$ and $\theta_B \in [\pi, 2\pi]$ the intersection of the regions r_1 and r_4 is a line. Therefore,

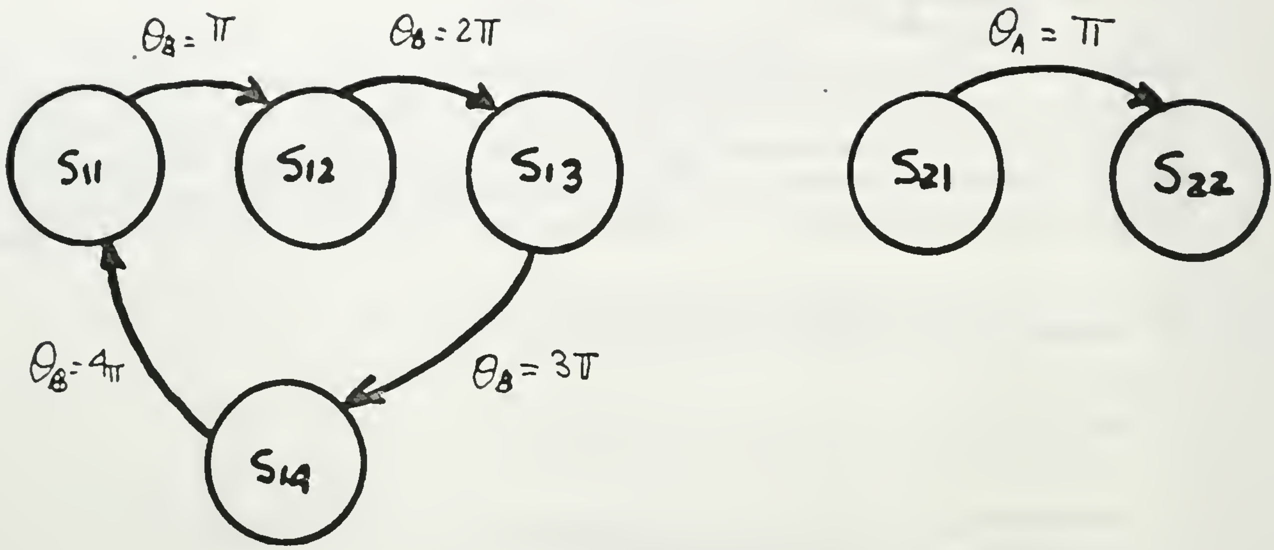


Figure 7.4: Two State Diagrams for the Half Gear Pair.

the least constrained region diagram $R(A, B)$ has three regions: r_3, r_2 and $r_1 \cap r_4$. This corresponds to the configuration space shown in the the bottom of Figure 7.5.

Assuming that S_1 and S_2 form a partial description of the pair's desired behavior, the configuration space $CO(A, B)$ corresponding to the pair of gears in Figure 7.1 matches $R(A, B)$ (here we are assuming that the configuration space boundary produced by teeth contact forms a line, instead of a narrow region as described in [Faltings87]). Thus, given the two state diagrams S_1 and S_2, and the initial shape of A as a full gear, the design algorithm will produce the half gear B.

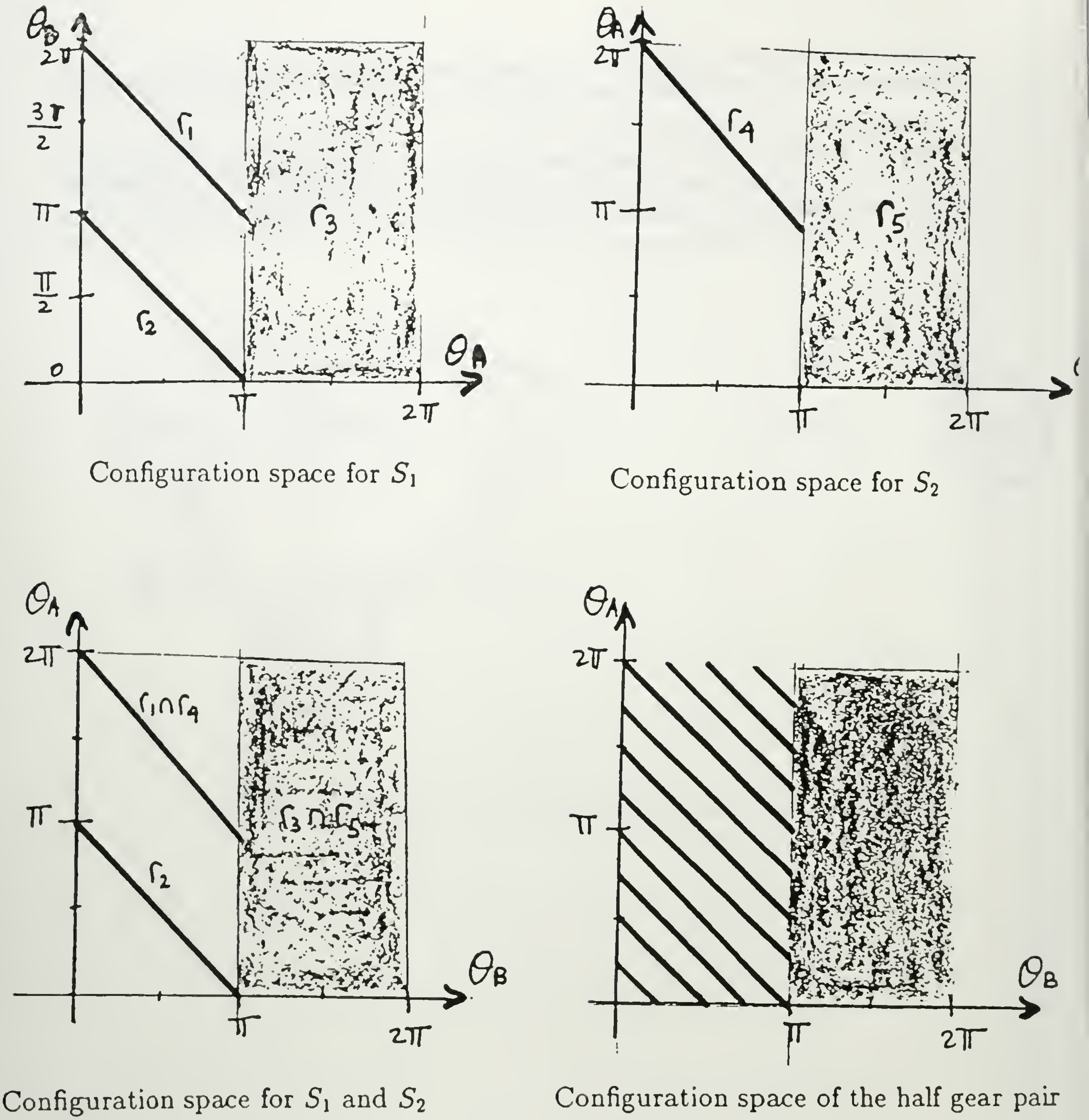


Figure 7.5: The Two Configuration Spaces Corresponding to the State Diagrams S_1 and S_2, their Composition, and the Actual Configuration Space of the Half Gear Pair. Shaded Regions Represent Free Positions.

8 Extensions

In this section, we will consider several extensions to the assumptions made in section 2. These extensions include:

- Three dimensional objects
- Several fixed axes for each object
- Three or four dimensional configuration spaces
- Movable axis pairs
- Multiple part mechanisms

Introducing three dimensional objects whose surfaces are either planar or cylindrical requires the consideration of new possible contacts between objects: vertex-face, edge-face, cylinder-face, etc. These contacts must be added to the elementary interactions table since they represent new possibilities for creating configuration space boundaries. In addition, the existing contacts (edge-edge, vertex-edge, etc.) must be reconsidered and augmented to reflect the third dimension. This introduces more parameters and equations for describing of the relation between features and configuration space boundaries, but does not alter the indeterminacy of the systems of equations. For example, if we consider again the vertex-edge contact in the translation-translation space (section 4.1), each vertex and point is determined by three coordinates. This brings to 9 the number of unknowns to be determined by 7 equations (since the dimensionality of the configuration space did not change, each contact produces 3 equations instead of 2). Therefore, determining one vertex in either A or B fully constraints the problem as it did in the case of two dimensions. The new cases involving a face (face-vertex, face-edge or face-face) do not introduce more indeterminacies since faces are treated as planes bounded by edges and vertices. In the case of the translation-translation space, the configuration space boundary produced by a face in contact with another feature is also a line, as all other contacts. This augments the number of possible feature contacts that have to be considered in order to obtain a boundary of $R(A, B)$. In general, new types of configuration space boundaries and more possible feature contacts producing the same type of configuration space boundary have to be added to the table of elementary interactions. From these considerations, we can conclude that the proposed approach does scale up for designing kinematic pairs containing three dimensional objects.

An interesting extension is to allow objects to have motions along several fixed axes. For example, we can require that object A translates along an axis O_1 and then rotate around O_2 (but not simultaneously); object B should translate along O_3 independently of A. For each pair of axes (one for each object), we can define a design problem (and its corresponding desired configuration space) that can be solved using the techniques presented here. The difficulty lies in the fact that proposed design solutions for each of the independently stated problems may conflict with each other. To overcome this difficulty, design processes that are independently solving the single axis problem must communicate and signal which object features can be deleted or added and what are the consequences of such changes. Object features having an active role in each configuration space are labeled as necessary. They will not be modified unless there is another alternative that replicates their effect. The backtracking over possible choices now occurs between choices in several configuration spaces, not just one. If the number of choices of additions and deletions is finite

for each configuration space, then a backtracking search algorithm will always find a solution, if such a solution exists.

Three or four dimensional configuration spaces arise only in fixed axis mechanisms where the objects are three dimensional. In this case the matching procedure has to compare and intersect two or three dimensional surfaces and three or four dimensional regions of space. The geometrical routines to carry these operations are much more complicated than the routines for the two dimensional case. This extension appears to be difficult and thus it seems better to try to work with several two-dimensional projections of the configuration space and treat them as a multiple fixed axis problem in different two-dimensional spaces.

Extending the design algorithm to deal with movable axis pairs is not possible in the context presented here since the analysis of their configuration space cannot be handled with the set of possible motion labels. For fixed axis pairs, we were able to identify three types of possible motions that defined a small number of design spaces. When the axes along which the objects in the pair move, this classification is no longer possible.

The ultimate goal of a design theory for mechanical devices is to show how complete (multiple part) mechanisms can be designed from a set of functional specifications. The kinematic design method proposed in this section is appropriate for the design of kinematic pairs, but is limited for designing complete mechanisms. An object generally belongs to several kinematic pairs and thus a modification in its shape affects several kinematic pairs, possibly modifying their configuration space. Therefore, we need to account for interactions between design tasks.

Following is a sketch of a procedure that designs a fixed axis mechanism by designing its kinematic pairs. The initial specification of the desired behavior must include the number of objects involved, the fixed axes for each object, and the pairwise kinematic behavior of every pair of objects that interact. For every kinematic pair, we set a design process and construct its desired configuration space. The interaction between design processes can be done as described in the case of multiple axes, by exchanging messages about the relevance of specific features of common objects. A constraint propagation technique is necessary to carry this information to all pairs that can be affected by a change.

This procedure has several limitations. First, it requires the determination of all the objects that will form the mechanism, together with their axes of motion. There is no possibility to introduce a new object, nor to change the orientation of an axis. Second, a detailed pairwise specification of the desired behavior is seldom available at the beginning of the design task. In general, the initial description of the desired behavior includes only a few objects and is refined as the detailed design proceeds, creating the constraints and specifications of each kinematic pair. Finally, this procedure is prohibitively expensive. Reasoning only from first principles to design a complete device is clearly inappropriate. Thus, a hierarchical approach for the design of complete mechanisms is necessary.

9 Implementation

We are in the process of implementing the general backtracking algorithm presented in section 3.3, initially for the translation-translation space. The analysis and computation of the translational configuration space has been fully implemented, as well as the component that compares two configuration spaces. Several examples of design in this space have been successfully tried out. We plan to extend the scope of the program to include all the other design spaces in the near future.

10 Comparison to Previous Work

Existing automated design systems in the domain of mechanical devices, are mostly parameter-based, and thus are not capable of innovative design of elementary components. Several approaches for routine design have been proposed. Dixon and Simmons [Dixon84] propose a generate-and-test approach in which a complete preliminary design is first performed, and then evaluated. If the design solution is not satisfactory, the process is iterated. The disadvantage of this method is that even the smallest failure in the design solution leads to the rejection of the whole solution, thus starting the design process from scratch. Mittal et al. [Mittal86a], [Mittal87b] and Brown and Chandrasekaran, [Brown86], propose a knowledge-based scheme that proceeds by hierarchical refinement of an initial generic device. At each refinement step, the constraints generated by the new requirements guide the design process. The final step consists of choosing the appropriate elementary components and the values of their parameters from a fixed library of components. Our innovative design method for kinematic pairs can be integrated with the last step of this refinement process, when elementary components (at the level of kinematic pairs) are chosen. Elementary components can be modified when their parametric description does not satisfy the design requirements.

First principles theories for reasoning about the shape and function of objects in mechanical devices have been developed recently. The idea of using configuration spaces as an intermediate representation that links object structure and kinematic behavior was independently proposed by Faltings and Forbus [Faltings86], [Faltings87], [Forbus87], and Joskowicz [Joskowicz87a], [Joskowicz87b], [Joskowicz87c]. This work has concentrated on the analysis of mechanical devices to produce a qualitative description of their kinematic behavior. The design method suggested in [Faltings87] uses configuration spaces but considers only the case of parameter variation. The design methods presented in this paper provide further evidence for the utility of configuration spaces as a first principles representation.

A recent paper, [Murthy87], describes a system (PROMPT), that is capable of reasoning about the structure and behavior of elementary components. Given a set of design constraints, the system is capable of modifying elementary components to match the given design constraints. The modifications are made using a set of *modification operators* that are capable of analyzing the structure of the component, determine its behavior and establish a mapping between the results of the analysis to the structural changes required to satisfy design constraints. The paper demonstrates these ideas for the domain of structural beam design, and shows how the shape of the beam cross section is modified to match design constraints. The shape of an initial beam is modified by applying shape operations such as mass redistribution, circular symmetry, etc. The precompilation of shape changes results in a level of reasoning that forms a bridge between the purely parametrized level of component description and the general, but inefficient, level of first principles analysis. Because the shapes of the prototypes are changed by analyzing their structure, this design approach falls

in the realm of innovative design. Its generality is determined by the scope of the modification operators. The object feature addition and deletion that we use in designing objects can be considered as two types of modification operators for kinematic pairs, although these operators are fully general since they are based on a first principles theory.

11 Conclusion

In this paper, we addressed the problem of designing the shape of physical objects defined by a set of functional requirements. In particular, we showed how to design elementary components of mechanical devices (kinematic pairs) from a description of their desired behavior. The novelty of our approach consists of using a first principles theory of kinematic behavior to reason about the relation between function and shape of objects. This theory, initially presented in a previous work [Joskowicz87], and used for the qualitative analysis of mechanical devices, introduces configuration spaces as an intermediate representation that links kinematic behavior and object geometry. In the present work, we show how this framework can be used to design kinematic pairs. The main advantage of the techniques described here is that they support a potentially infinite range of object's shape creation and modifications, thus overcoming the limitations of parametrized approaches in which this range is limited by the number of parameters describing each shape.

In our approach, the desired kinematic behavior of the pair is initially specified either in terms of possible motions, or causally. Such descriptions are mapped into an equivalent desired configuration space. According to the types of desired motions specified for each object, one of five design spaces is selected (translation-translation translation-rotation, etc.). When the objects have an initial shape, their actual configuration space is compared with the desired configuration space. The differences between these two configuration spaces are recorded, and then reduced by adding and deleting features (edges, arcs, etc.) to the objects. To determine which features to add and delete, a table of elementary interactions describing the configuration space boundaries created by feature contacts (edge-vertex, arc-vertex, etc.) is used. The design process is considered successful when the actual configuration space of the modified pair is identical to the desired configuration space that represents the intended behavior.

We have provided a number algorithms for shape modification. The first algorithm is a design space independent backtracking algorithm that examines the possible choices that are available to modify the object shapes. When one of the objects has a fixed shape (i.e. no changes to it are allowed), or when two disconnected object features are not introduced simultaneously, this algorithm always finds a solution, if such a solution exists. The complexity of the algorithm is exponential in the size of the desired configuration space. This algorithm has been extended to the case in which the desired behavior is described qualitatively, and thus the precise configuration space boundaries is not available. We have also studied the translation-translation design space in detail, and provided a number of polynomial-time algorithms for this space. Figure 11.1 presents a table summarizing these results.

	Precise Boundary			Qualitative Boundary	
	Topology	**Translation-Translation**	**Rotation Spaces**	**Translation-Translation**	**Rotation Spaces**
ONE	Both Objects Convex	Algorithm 5.1 $O(n+m)$	Algorithm 4.1 $O(nm)$	As in the precise bounadry case.	Limited Backtracking No Fill-in necessary
SHAPE	One Object Convex	Algorithm 5.3 or 5.4 $O((n+m)\log(n+m))$	Bactracking Algorithm (Complete)	Idem	Bactracking Algorithm (Not Complete)
FIXED	None Convex	Algorithm 5.5 $O(n^2k\log n^2k)$	+ Fill-in Procedure	Idem	+ Fill-in Procedure
NO	Both Objects Convex	Algorithm 5.2, $O(n+m)$ or Limited Backtracking	Algorithm 4.1, $O(nm)$ Complete	Idem	Limited Backtracking Not Complete
SHAPE	One Object Convex	Backtracking	Algorithm	Backtracking	Algorithm
FIXED	None Convex	(Not	Complete)	(Not	Complete)

Figure 11.1 Summary of Results.

12 References

[Brown86] "Knowledge and Control for a Mechanical Design Expert System", Brown, D. and Chandrasekaran, B. in *Computer*, July 1986

[DeKleer84] "A Qualitative Physics based on Confluences" DeKleer, J. and Brown, J. in *Artificial Intelligence* 24, 1984.

[Dixon84] "An Architecture for Application of Artificial Intelligence to Design", Dixon J., Simmons M. and Cohen P. in *Proc. ACM/IEEE 21st Design Automation Conf.*, June 1984

[Dixon86] "Artificial Intelligence and Design: A Mechanical Engineering View", Dixon J. in *Proceedings of the Fifth AAAI Conference*, Philadelphia 1986.

[Encarnacao83] *Computer-Aided Design* Encarnacao, J. and Schlechtendal, E. Springer-Verlag , Berlin, 1983.

[Faltings86] "A Theory of Qualitative Kinematics in Mechanisms", Faltings, B. in Report UIUCDCS-R-86-1274, University of Illinois, May 1986.

[Faltings87a] "Qualitative Place Vocabularies for Mechanisms in Configuration Space", Faltings, B. in Phd. Dissertation, Report No. UIUCDCS-R-87-1360, University of Illinois, July 1987.

[Faltings87b] "Qualitative Kinematics in Mechanisms" Faltings, B. in *Proceedings of IJCAI-87*, Milano, Italy, 1987.

[Forbus84] "Qualitative Process Theory" Forbus, K. in *Artificial Intelligence* 24, 1984.

[Forbus87] "The Inferential Structure of Qualitative Kinematics", Forbus K., Nielsen P. and Faltings B. in *Proceedings of IJCAI-87*, Milano, Italy, 1987.

[Joskowicz87a] "A Framework for the Kinematic Analysis of Mechanical Devices", Joskowicz L. in Technical Report 313, Computer Science Department, Courant Institute, New York University, August 1987.

[Joskowicz87b] "Shape and Function in Mechanical Devices", Joskowicz L. in *Proceedings of the Sixth National Conference on Artificial Intelligence*, Seattle, 1987.

[Joskowicz87c] "Reasoning about the Kinematics of Mechanical Devices", Joskowicz L. to appear in *Artificial Intelligence in Engineering*, 1988.

[Kuipers84] "Commonsense Reasoning about Causality: Deriving Behavior from Structure", Kuipers, B. in *Artificial Intelligence* 24, 1984.

[Lozano-Pérez79] "An Algorithm for Planning Collision-Free Paths among Polyhedral Obstacles", Lozano-Pérez, T. and Wesley, M. in *Communications of the ACM*, Vol. 22, 1979.

[Lozano-Pérez82] "Spatial Planning: A Configuration Space Approach", Lozano-Pérez, T. in *IEEE Transactions on Computers*, Vol C-32, No. 2, 1982.

[Mitchell85] "A Knowledge-Based Approach to Design" Mitchell T., Stenberg L., Shulman J. in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, September 1985.

[Mittal86a] "PRIDE: An Expert System for the Design of Paper Handling Systems" Mittal S., Dym C. and Morjaria M. in *Computer*, July 1986

[Mittal86b] "A Knowledge-Based Framework for Design", Mittal S. and Araya A. in *Proceedings of the Sixth AAAI Conference*, Philadelphia 1986.

[Murthy87] "PROMPT: An Innovative Design Tool", Murthy S. and Addanki S. in *Proceedings of the Sixth AAAI Conference*, Philadelphia 1986.

[Nielsen87] "The Qualitative Statics of Rigid Bodies", Nielsen P. in Report No. UIUCDCS-R-87-1354, University of Illinois, July 1987.

[Preparata85] *Computational Geometry: An Introduction*, F. Preparata and M. Shamos, *Springer-Verlag Editors*, 1985

[Reuleaux76] *The Kinematics of Machinery: Outline of a Theory of Machines*, Reuleaux F., 1876, (Reprinted by Dover Publications Inc., 1963).

[Schwartz83] "On the Piano Movers II. General Techniques for Computing Topological Properties on Real Algebraic Manifolds", Schwartz, J.T. and Sharir M. in *Advances in Applied Mathematics 4*, 1983.

13 Appendix

In this appendix, we analyze in detail some of the aspects of object contacts and the configuration space boundaries they create. We begin by presenting our notation which follows [Faltings87]. The first section provides the equations of a point in configuration space created by the contact of two points in the object boundary. The second section present the equations for both analysis and design in the translation-translation space (includes arc segments), and an example of contact in the rotation-rotation space. We also discuss the conditions for qualitative design. This appendix is based on the analysis results presented in [Faltings87], and extends the results for design.

Let $\mathbf{X_A}$, $\mathbf{X_B}$, $\theta_\mathbf{A}$, $\theta_\mathbf{B}$ be the motion parameters of A and B and let O_A and O_B be their axes of motion. Let $\mathbf{F}$ be the configuration space coordinate frame that is fixed in space. $\mathbf{F}$ is defined by two motion parameters and an origin $P_0 = (X_A^0, X_B^0)$. A point in the configuration space F is denoted by $P_C = (X_A, X_B)$, where X_A and X_B are the values of the motion parameters $\mathbf{X_A}, \mathbf{X_B}$ with respect to the origin P_0. Let $F_A = (\mathbf{x_A}, \mathbf{y_A})$ and $F_B = (\mathbf{x_B}, \mathbf{y_B})$ be two cartesian coordinate frames attached to A and B, respectively. Object features are described in terms of coordinates with respect to the *local* coordinate frames (see Figure A.1)

- Points of objects are denoted by $P_A = (x_A, y_A)$ and $P_B = (x_B, y_B)$.
- Edges of A and B are defined by two endpoints, $L_A = (P_1^A, P_2^A)$ and $L_B = (P_1^B, P_2^B)$.
- Arcs of objects are defined by four values: the origin point o_A, the radius of the arc r_A and the two endpoints of the arc. P_1^A and P_2^A.

We measure angular positions with respect to the $\mathbf{x_A}$ axis. The angular position of the object is given by θ_A, and ψ_A represents the local angular offset of a feature with respect to $\mathbf{x_A}$. Similarly, we measure translations with respect to the origin of the $\mathbf{x_A}$ axis, and parallel to it. The translational position of the object is given by X_A, and Δ_A represents the local translational offset of a feature with respect to the origin of $\mathbf{x_A}$ (see Figure A.1).

We define the measure of distance between the axes O_A and O_B as follows (see Figure A.2):

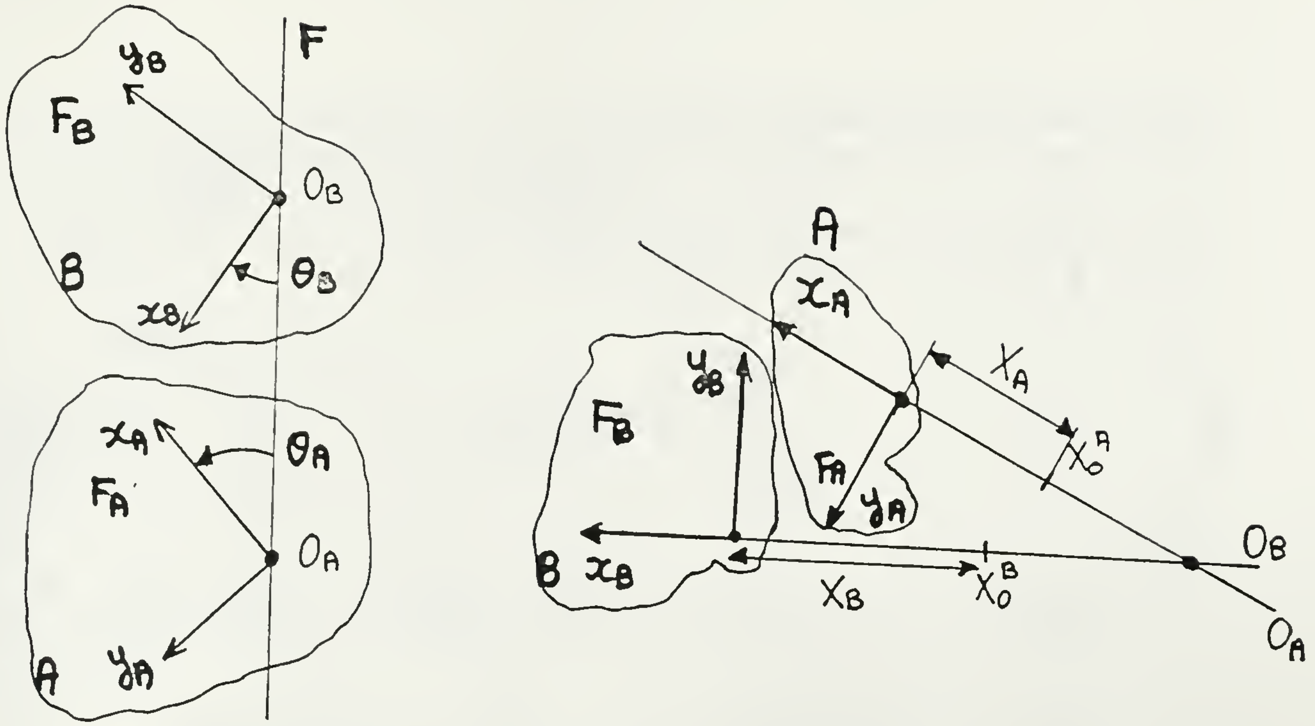
- Rotation-Rotation Space: The distance d between O_A and O_B is denoted by d.
- Rotation-Translation Space: The distance between O_A and a perpendicular to O_B. The value l is the signed distance from the perpendicular to the origin of O_B.
- Translation-Translation: The angle between the two axes is α. The values l_A and l_B are the signed distances from the origin of O_A and O_B to the intersection point of the two axes, respectively.

In addition, we denote by R_A the distance from the point of rotation to the point of contact (rotation space) or from the point of contact to the axis of translation.

Without loss of generality, we assume that the origin of the local coordinate axes coincides with the axis of rotation, and the axis of translation coincides with the $\mathbf{x_A}$ axis, as shown in Figure A.2.

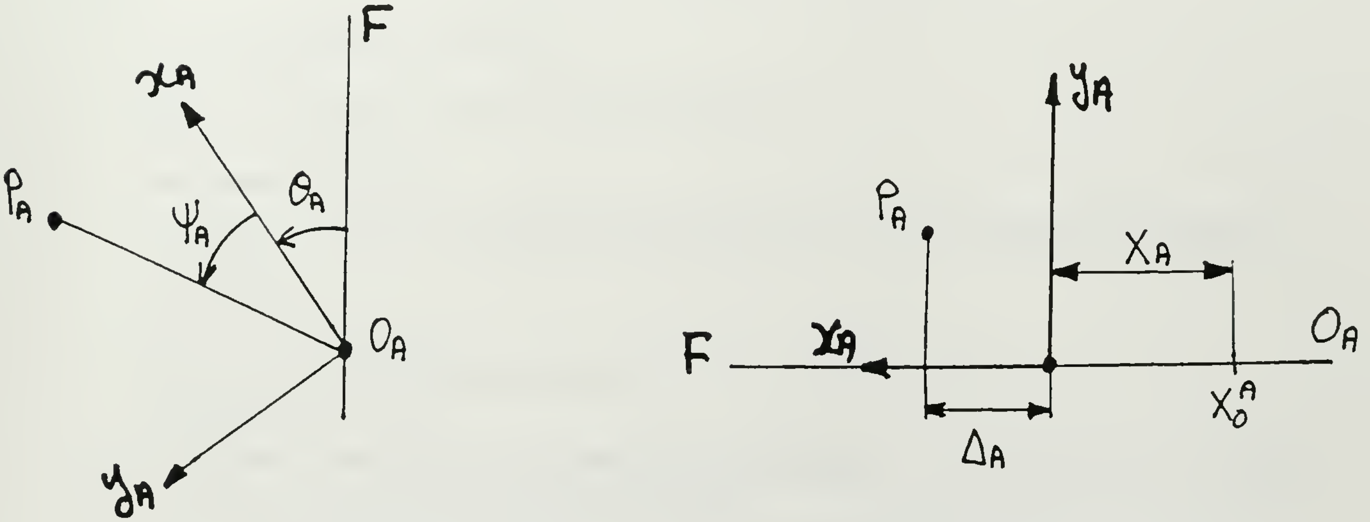
A Coordinates of Endpoints

Given an endpoint P_C in the configuration space boundary, our goal is to find the corresponding two points P_A and P_B of A and B that create it. These points can be vertices (and thus form a Type 0



Rotation-Rotation Translation-Translation

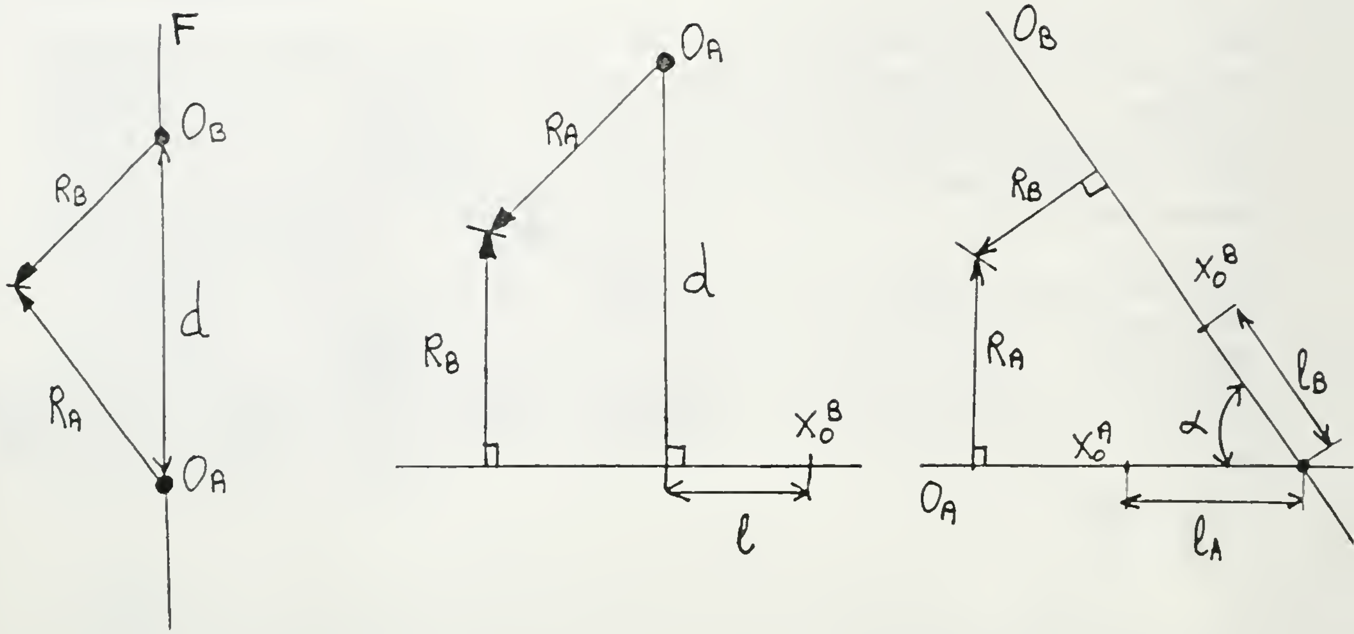
(a) Global and Local Coordinate Frames



Angular Offset Translational Offset

(b) Offset Measures

Figure A.1: Notation Conventions



Constant Axis Parameters

Figure A.2: Notation Conventions (continued).

contact), or just points belonging to the boundary features. The problem of determining P_A and P_B from P_C is underconstrained, and thus accepts infinitely many solutions. However, if one of the points P_A or P_B are given, the other is completely determined.

In this section, we provide the equations relating P_A, P_B and P_C. They can be used for both analysis and design.

A.1 Rotation-Rotation

We distinguish two cases: $R_A + R_B \geq d$ and $R_A + R_B \leq d$.

1. $R_A + R_B \geq d$. The general equations are:

$$R_A \cos(\theta_A + \psi_A) + R_B \cos(\theta_B + \psi_B) = d \qquad (1)$$
$$R_A \sin(\theta_A + \psi_A) = R_B \cos(\theta_B + \psi_B) \qquad (2)$$

$$\tan \psi_A = x_A / y_A \qquad (3) \qquad\qquad \tan \psi_B = x_B / y_B \qquad (5)$$
$${R_A}^2 = {x_A}^2 + {y_A}^2 \qquad (4) \qquad\qquad {R_B}^2 = {x_B}^2 + {y_B}^2 \qquad (6)$$

From (1) and (2) we deduce

$$R_A = \frac{d}{\cos(\theta_A + \psi_A) + \sin(\theta_A + \psi_A)\cot(\theta_B + \psi_B)} \qquad (7)$$

$$R_B = \frac{d}{\cos(\theta_B + \psi_B) + \sin(\theta_B + \psi_B)\cot(\theta_A + \psi_A)} \qquad (8)$$

For analysis, we have the following:

Given: P_A, P_B (we can then deduce R_A, R_B, ψ_A, ψ_B).

Find: $P_C = (\theta_A, \theta_B)$.

Equations: (1) - (6)

Problem: Two unknowns, eight given values, six equations. The problem is fully determined.

For design,

Given: $P_C = (\theta_A, \theta_B)$.

Find: P_A, P_B (and also R_A, R_B, ψ_A, ψ_B).

Equations: (1) - (6).

Problem: Eight unknowns, two given values, six equations. The problem is underconstrained. One point (either P_A or P_B) must be given.

Assume P_A is given. Then, we get from equations (1) - (6) and (8):

$$\frac{d}{R_A} = \cos(\theta_A + \psi_A) + \sin(\theta_A + \psi_A)\cot(\theta_B + \psi_B)$$

where R_A and ψ_A are given by (3) and (4). Then,

$$\tan(\theta_B + \psi_B) = \frac{\sin(\theta_A + \psi_A)}{\frac{d}{R_A} - \cos(\theta_A + \psi_A)} = C$$

By the tangent rule, we get,

$$\tan \psi_B = \frac{\tan \theta_B - C}{1 - C \tan \theta_B} \tag{9}$$

Thus, the point P_B is defined by

$$x_B = \pm \frac{R_B \tan \psi_B}{\sqrt{1 + \tan^2 \psi_B}} \tag{10}$$

$$y_B = \pm \frac{R_B}{\sqrt{1 + \tan^2 \psi_B}} \tag{11}$$

where ψ_B and R_B are computed by equations (9) and (7). If $0 \leq \psi_A + \theta_A \leq \pi$ then $x_B = +$ and $y_B = -$, else $x_B = -$ and $y_B = -$.

The case where P_B is given is entirely symmetrical. Just exchange the subindices containing A to subindices containing B.

Special Case:

A special case occur when (7) and (8) have a denominator equal to zero. In this case,

$$\tan(\theta_A + \psi_A) = -\tan(\theta_B + \psi_B)$$

Then, $\theta_A + \psi_A = [\theta_B + \psi_B + \pi]_{mod_2\pi}$. Note that this contact is physically infeasible. Therefore, the values of ψ_A and ψ_B for which the above equation is true must not be considered.

2. $R_A + R_B \leq d$. This condition corresponds to a concave contact. The general equations are:

$$R_A \cos(\theta_A + \psi_A) - R_B \cos(\theta_B + \psi_B) = d \tag{1}$$

$$R_A \sin(\theta_A + \psi_A) = R_B \cos(\theta_B + \psi_B) \tag{2}$$

Equations (3) - (6) are the same as before. We get,

$$R_A = \frac{d}{-\cos(\theta_A + \psi_A) + \sin(\theta_A + \psi_A)\cot(\theta_B + \psi_B)} \tag{7}$$

$$R_B = \frac{d}{-\cos(\theta_B + \psi_B) + \sin(\theta_B + \psi_B)\cot(\theta_A + \psi_A)} \tag{8}$$

The rest of the equations are exactly identical to the case $R_A + R_B \geq d$.

A.2 Rotation-Translation

No special cases to distinguish (d or l can be zero). The general equations are:

$$R_A \cos(\theta_A + \psi_A) + R_B = d \qquad (1)$$
$$R_A \sin(\theta_A + \psi_A) = (\Delta_B + X_B) - l \qquad (2)$$

$$\tan\psi_A = x_A/y_A \qquad (3) \qquad\qquad y_B = R_B \qquad (5)$$
$$R_A{}^2 = x_A{}^2 + y_A{}^2 \qquad (4) \qquad\qquad x_B = \Delta_B \qquad (6)$$

From (1) and (2) we deduce that

$$R_B = d - (\Delta_B + X_B - l)\cot(\theta_A + \psi_A) \quad (7) \qquad\qquad R_A = \frac{\Delta_B + X_B - l}{\sin(\theta_A + \psi_A)} \qquad (8)$$

For analysis, we have the following

Given: P_A, P_B (we can then deduce $R_A, R_B, \psi_A, \Delta_B$).

Find: $P_C = (\theta_A, X_B)$.

Equations: (1) - (6).

Problem: Two unknowns, eight given values, six equations. The problem is fully determined.

For design,

Given: $P_C = (\theta_A, X_B)$.

Find: P_A, P_B (and also $R_A, R_B, \psi_A, \Delta_B$).

Equations: (1) - (6).

Problem: Eight unknowns, two given values, six equations. The problem is underconstrained. One point (either P_A or P_B) must be given.

Assume P_A is given. Then, we get from equations (1) - (6) and (8):

$$\Delta_B = R_A \sin(\theta_A + \psi_A) - X_B + l$$

$$R_B = d - R_A \cos(\theta_A + \psi_A)$$

where R_A and ψ_A are given by (3) and (4). Thus, the point P_B is defined by

$$x_B = R_A \sin(\theta_A + \psi_A) - X_B + l \qquad (9)$$
$$y_B = d - R_A \cos(\theta_A + \psi_A) \qquad (10)$$

Assume now that P_B is given. Then Δ_B and R_B are deduced from (5) and (6). From (8) we get:

$$\tan(\theta_A + \psi_A) = \frac{\Delta_B + X_B - l}{d - R_B} = C$$

By the tangent rule,

$$\tan\psi_A = \frac{\tan\theta_A - C}{1 - C\tan\theta_A} \tag{9}$$

Thus, the point P_A is defined by

$$x_A = \pm\frac{R_A\tan\psi_A}{\sqrt{1+\tan^2\psi_A}} \quad (10) \qquad\qquad y_A = \pm\frac{R_A}{\sqrt{1+\tan^2\psi_A}} \quad (11)$$

where ψ_A and R_A are computed by equations (9) and (7). If $0 \leq \theta_A + \psi_A \leq \pi$ then $x_B = +$ and $y_B = -$, else $x_B = -$ and $y_B = -$.

Special Case

A special case occurs when (9) has a denominator equal to zero. Then,

$$\theta_A = \arctan(\frac{d - x_B}{y_B + X_B - l})$$

In this case, the contact is physically infeasible. Therefore, the points P_B that satisfy the above equation should not be considered as solutions.

A.3 Translation-Translation

We distinguish three cases: First, when $\alpha = 0$, then when $\alpha = \pi/2$, and finally the general case where $\alpha \neq 0$ and $\alpha \neq \pi/2$.

1. $(\alpha \neq 0) \wedge (\alpha \neq \pi/2)$. The general equations are:

$$\frac{R_A}{\cos\alpha} + R_B = (X_B + \Delta_B - l_B)\tan\alpha \tag{1}$$
$$\frac{R_B}{\cos\alpha} + R_A = (X_A + \Delta_A - l_A)\tan\alpha \tag{2}$$

$$x_A = \Delta_A \quad (3) \qquad\qquad x_B = \Delta_B \quad (5)$$
$$y_A = R_A \quad (4) \qquad\qquad y_B = R_B \quad (6)$$

For analysis,

Given: P_A, P_B (we can then deduce $R_A, R_B, \Delta_A, \Delta_B$).

Find: $P_C = (X_A, X_B)$.

Equations: (1) - (6).

Problem: Two unknowns, eight given values, six equations. The problem is fully determined.

For design,

Given: $P_C = (X_A, X_B)$.

Find: P_A, P_B (and also $R_A, R_B, \Delta_A, \Delta_B$).

Equations: (1) - (6).

Problem: Eight unknowns, two given values, six equations. The problem is underconstrained. One point (either P_A or P_B) must be given.

Assume P_A is given. Then, we get from equations (1) - (6) and (8):

$$y_B = (X_A - x_A - l_A)\tan\alpha - y_A \cos\alpha \tag{7}$$

$$x_B = \frac{y_A + y_B \cos\alpha}{\tan\alpha} + l_A - X_A \tag{8}$$

The case where P_A is given is exactly symmetrical. There are no special cases.

2. $\alpha = 0$ Let d be the distance between the two parallel axes. The general equations are:

$$R_A + R_B = d \tag{1}$$

$$X_A + \Delta_A - l_A = X_B + \Delta_B - l_B \tag{2}$$

Equations (3) - (6) are as above. Assuming P_A is given, we have:

$$x_B = (x_A + X_A) - X_B + (l_B - l_A) \tag{7}$$

$$y_B = d - y_A \tag{8}$$

When P_B is given the equations are exactly symmetrical.

3. $\alpha = \pi/2$ The general equations are:

$$R_A = (\Delta_B + X_B) - l_B \tag{1}$$

$$R_B = (\Delta_A + X_A) - l_A \tag{2}$$

Equations (3) - (6) are as above. Assuming P_A is given, we have:

$$x_B = y_A + l_B - X_B \tag{7}$$

$$y_B = x_A + l_A - X_A \tag{8}$$

When P_B is given the equations are exactly symmetrical.

B Configuration Space Boundary Equations

In this section we provide the configuration space equations resulting from two feature contacts. There are nine possible contacts, and three spaces to analyze. Note that the case of a vertex-vertex contact was already analyzed in the previous section. It produces a point in configuration space.

For each case we will present the equation created by the contact (analysis) and the appropriate equalities for the design case. In addition, we analyze the conditions necessary to maintain monotonicity, so that qualitative relations hold.

B.1 Translation-Translation

For the sake of simplicity, we assume that one object is fixed and the other has two degrees of translational freedom. This is equivalent to having the two objects with one degree of translational freedom each, where the axis are perpendicular ($\alpha = \pi/2$).

Vertex-Vertex Contact

Two vertices P_A and P_B create a configuration space point P_C. This is a Type 0 boundary.

Analysis: $P_C = (P_A - P_B) + P_0$

Design: Underconstrained. There are no special cases. Fixing one feature determines the other.
Given P_A, we get $P_B = (P_A - P_C) + P_0$
Given P_B, we get $P_A = (P_B + P_C) - P_0$

Edge-Vertex Contact

An edge $L_A = (P_1^A, P_2^A)$ and a vertex P_B create a configuration space line segment $L_C = (P_1^C, P_2^C)$. This is a Type 1 boundary. L_A and L_C are parallel.

Analysis: $P_1^C = (P_1^A - P_B) + P_0$
$P_2^C = (P_2^A - P_B) + P_0$

Equation of the line: $(y_1^A - y_2^A)\mathbf{X_A} + (x_2^A - x_1^A)\mathbf{X_B} + y_2^A x_1^A - y_1^A x_2^A = 0$

Design: Underconstrained (Five equations and six unknowns). There are no special cases. Fixing one feature determines the other two.

Given P_B, we get $P_1^A = (P_B - P_1^C) - P_0$
$P_2^A = (P_B - P_2^C) - P_0$

Given P_1^A, we get $P_B = (P_1^A - P_1^C) + P_0$
$P_B = (P_{12}^A - P_2^C) + P_0$

Similarly when P_2^A is given.

Vertex-Edge Contact

Exactly as above, but reversing the roles of L_A and P_B.

Edge-Edge Contact

An edge $L_A = (P_1^A, P_2^A)$ and an edge $L_B = (P_1^B, P_2^B$ create a configuration space line segment $L_C = (P_1^C, P_2^C)$. This is a Type 1 boundary. All L_A, L_B and L_C are parallel.

Analysis: $P_1^C = (P_2^A - P_1^B) + P_0$
$P_2^C = (P_1^A - P_2^B) + P_0$

All the lines are parallel:

$$\frac{y_2^A - y_1^A}{x_2^A - x_1^A} = \frac{y_2^B - y_1^B}{x_2^B - x_1^B} = \frac{y_2^C - y_1^C}{x_2^C - x_1^C}$$

Equation of the line: $(y_1^A - y_2^A)\mathbf{X_A} + (x_2^A - x_1^A)\mathbf{X_B} + y_2^A x_1^A - y_1^A x_2^A = 0$

Design: Underconstrained (Five equations and eight unknowns). There are no special cases. Fixing two features determines the other two.

Given two endpoints of L_A and L_B, P_1^A and P_1^B, we get
$$P_2^A = (P_1^C + P_1^B) - P_0$$
$$P_2^B = (P_1^A - P_2^C) + P_0$$

Similarly when any other combination of two endpoints, one from L_A and one from L_B.

Given one edge $L_A = (P_1^A, P_2^A)$, we get
$$P_1^B = (P_2^B + P_2^C) - P_0$$
$$P_2^B = (P_1^B + P_1^C) - P_0$$

Similarly when E_B is given.

The only special case is when the edges are horizontal, i.e. $x_1^A - x_2^A = x_1^B - x_2^B = x_1^C - x_2^C$. Here again, two endpoints determine the other two.

Vertex-Arc Contact

A vertex P_A and an arc $L_B = (o_B, r_B, P_1^B, P_2^B)$ create a configuration space arc segment $L_C = (o_C, r_C, P_1^C, P_2^C)$. This is a Type 2 boundary. The center of the new arc is the same as the center o_B, as well as the radius. This is true for both convex and concave arcs

Analysis: $P_1^C = (P_A - P_1^B) + P_0$
$P_2^C = (P_A - P_2^B) + P_0$

The radiuses are equal, and the origins coincide:

$$r_C = r_B = \sqrt{(x_1^B - x_0^C)^2 + (y_1^B - y_0^C)^2} = \sqrt{(x_1^B - x_0^C)^2 + (y_1^B - y_0^C)^2}$$
$$o_C = o_B + P_0$$

Equation of the arc: $(\mathbf{X_A} - x_0^C)^2 + (\mathbf{X_B} - y_0^C)^2 = r_C{}^2$

Design: Underconstrained (Seven equations and nine unknowns). There are no special cases. Fixing one endpoint determines the other features.

Given the vertex P_A, we get the arc L_B as follows:
$$P_1^B = (P_A - P_1^C) - P_0$$
$$P_2^B = (P_A - P_2^C) - P_0$$
$$r_B = r_C$$
$$o_B = o_C - P_0$$

Similarly, when one of the enpoints of L_B is given, the other features are determined.

There are no special cases. One feature determines the others.

Qualitative Design:

In this case, only P_1^C and P_2^C are given. The minimum radius of both r_C and r_B is:

$$r_B = r_C \geq \sqrt{(x_1^C - x_2^C)^2 + (y_1^C - y_2^C)^2}/2$$

In order to have the boundary monotonically increasing or decreasing, we have the further restriction,

$$r_B = r_C \geq \sqrt{(x_1^C - x_2^C)^2 + (y_1^C - y_2^C)^2}/\sqrt{2}$$

There are now 12 unknowns and 9 equations. Thus, in order to fix the features of A and B we need

- Fix the vertex P_A and the radius r_B.
- Fix one endpoint of L_B and the radius r_B.
- Use the tangents of the adjacent features to determine the radius. Let α_1 and α_2 the angles of these tangents:

$$r_C \geq \sqrt{\frac{(x_1^C - x_2^C)^2 + (y_1^C - y_2^C)^2}{2(1 - \cos(\alpha_1 + \alpha_2))}}$$

guarantees a monotonic boundary.

Arc-Vertex

Symmetrical to the previous case.

Edge-Arc

An edge $L_A = (P_1^A, P_2^A)$ and an arc $L_B = (o_B, r_B, P_1^B, P_2^B)$ create a configuration space that is either empty or that is a line (see Figure A.3). When the arc is concave, the part of the edge that is in contact with it is the endpoint, and thus forms a vertex-arc contact. The only case in which a point of L_A that is not an endpoint touches L_B is when L_B is convex. In this case, a configuration space line segment L_C is produced. This is a Type 1 boundary. L_C is parallel to L_A.

Condition for the contact to take place: the tangent at the enpoints of L_B, at P_1^B must be greater than β, and at P_2^B must be smaller than β. β is the slope of the line L_A Otherwise, it is the endpoint P_1^B or P_2^B that is in contact. Let

$$\tan\beta = \frac{y_1^A - y_2^A}{x_1^A - x_2^A}$$

be the slope of the edge L_A.

Analysis (edge vs. concave arc):

$$\begin{aligned} x_1^C &= x_1^A + r_B \sin\beta + x_B + x_0 \\ y_1^C &= P_2^A + r_B \cos\beta + y_B + y_0 \end{aligned}$$

Parallelism condition:

$$\tan\beta = \frac{y_1^C - y_2^C}{x_1^C - x_2^C}$$

The point of contact, where $P_1^B = P_2^B$ is:

$$x_1^B = r_B \sin\beta + x_0$$
$$y_c^C = r_B \cos\beta + y_0$$

Design: Underconstrained (Six equations and twelve unknowns). The points P_1^B and P_2^B cannot be determined in this case. Only when the next edge-vertex or vertex-arc contact takes place we can assign them values. At this point, we can determine o_B and r_B, P_1^A and P_2^A.

- Fixing P_1^A and P_2^A determines o_B and r_B.
- Fixing o_B and r_B fixed P_1^A and P_2^A.

Arc-Edge Contact

Symmetrical to the previous case.

Arc-Arc Contact

An arc $L_A = (o_A, r_A, P_1^A, P_2^A)$ and an arc $L_B = (o_B, r_B, P_1^B, P_2^B)$ create a configuration space of Type 2 (arc): $L_C = (o_C, r_C, P_1^C, P_2^C)$. For such a contact to take place, both arcs must be convex, or one arc concave and one convex; the radius of the convex arc must be smaller than the radius of the convex arc (see Figure A.3). The center of the new arc coincides with the center of the fixed arc. The radius is either the sum of radius, or the substraction of them (second case). We assume that the contacts start and terminate at the enpoints of the arc.

Analysis: $P_1^C = (P_2^A - P_1^B) + P_0$
$P_2^C = (P_1^A - P_2^B) + P_0$

Other conditions: $r_C = r_A + r_B$ or $r_C = r_A - r_B$
$o_C = o_A + P_0$

$$r_C = \sqrt{(x_1^C - x_0^C)^2 + (y_1^C - y_0^C)^2} = \sqrt{(x_2^C - x_0^C)^2 + (y_2^C - y_0^C)^2}$$

$$r_B = \sqrt{(x_1^B - x_0^B)^2 + (y_1^B - y_0^B)^2} = \sqrt{(x_2^B - x_0^B)^2 + (y_2^B - y_0^B)^2}$$

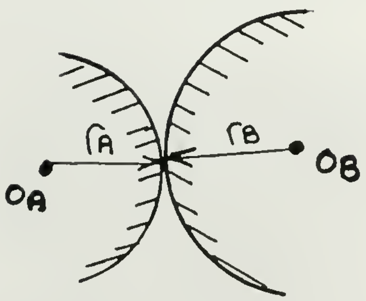
$$r_A = \sqrt{(x_1^A - x_0^A)^2 + (y_1^A - y_0^A)^2} = \sqrt{(x_2^A - x_0^A)^2 + (y_2^A - y_0^A)^2}$$

Design: Underconstrained (11 equations and 14 unknowns). Fixing two features determines the other two.

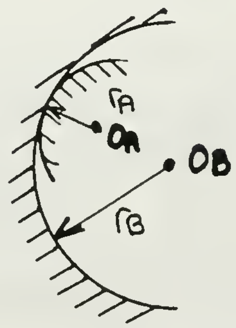
- Fix the radius r_A and an endpoint P_1^A.
- Fix two radiuses.
- Fix two centers.

The qualitative analysis is similar to the previous case.

(a) Edge-Arc contacts considered as Arc-Vertex or Edge-Vertex contacts.



Two conxex arcs



A convex and a concave arc

Figure A.3: Cases of Contacts Involving Arcs.

NYU COMPSCI TR-356 c.1
Joskowicz, Leo
Innovative shape design

NYU COMPSCI TR-356 c.1
Joskowicz, Leo
Innovative shape design

This book may be kept JUN 10 1988

FOURTEEN DAYS

A fine will be charged for each day the book is kept overtime.

GAYLORD 142 PRINTED IN U.S.A.